DOCUMENT RESUME

ED 285 570 IR 012 786

TITLE	1985 Annual Technical Report: A Research Program in Computer Technology. July 1984--June 1985.
INSTITUTION	University of Southern California, Marina del Rey. Information Sciences Inst.
SPONS AGENCY	Advanced Research Projects Agency (DOD), Washington, D.C.
REPORT NO	ISI/SR-86-170
PUB DATE	Dec 86
CONTRACT	MDA-903-81-C-0335
NOTE	157p.
PUB TYPE	Reports - Descriptive (141) -- Reports - Research/Technical (143)
EDRS PRICE	MF01/PC07 Plus Postage.
DESCRIPTORS	Computer Graphics; *Computer Networks; *Computer Oriented Programs; *Computer Software; *Expert Systems; *Man Machine Systems; Programing; Programing Languages; Research Projects; *Telecommunications

ABSTRACT

Summaries of research performed by the Information Sciences Institute at the University of Southern California for the U.S. Department of Defense Advanced Research Projects Agency in 17 areas are provided in this report: (1) Common LISP framework, an exportable version of the Formalized Software Development (FSD) testbed; (2) Explainable Expert Systems, a framework for building expert systems; (3) Formalized Software Development, a new automated software development paradigm; (4) Mappings, transformations for converting specifications of programs into implementations; (5) Command Graphics, a device-independent graphics system; (6) Internet Concepts Research, protocols for interconnection of computer communication networks; (7) Strategic Computing Information System, an automatic and efficient system for information exchange; (8) Wideband Communication, technology to support satellite-based wideband computer communication; (9) KITSERV, cost-effective test devices; (10) VLSI, a low-cost, fast turnaround LSI/VLSI device fabrication service; (11) Advanced VLSI; (12) Text Generation for Strategic Computing, methods for autonomous creation of text by machine; (13) Computer Research Support, operation and support of computing facilities and equipment; (14) Exportable Workstation Systems, remote testbed environment of advanced workstations and servers; (15) New Computing Environment, the next generation of computers and facilities; (16) Strategic Computing-Development Systems; and (17) Strategic Command, Control, and Communication Experiment Support. References are listed throughout. (Author/MES)

****
* Reproductions supplied by EDRS are the best that can be made *
* from the original document. *
****

ED285570

U.S. DEPARTMENT OF EDUCATION
Office of Educational Research and Improvement
EDUCATIONAL RESOURCES INFORMATION CENTER (ERIC)

☑ This document has been reproduced as received from the person or organization originating it.

☐ Minor changes have been made to improve reproduction quality.

- Points of view or opinions stated in this document do not necessarily represent official OERI position or policy.

ISI/SR-86-170

University
of Southern
California

1985

ANNUAL TECHNICAL REPORT

July 1984 - June 1985

A Research Program in Computer Technology

IR013786

INFORMATION
SCIENCES
INSTITUTE
ISI

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

BEST COPY AVAILABLE

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION
Unclassified

1b. RESTRICTIVE MARKINGS

2a. SECURITY CLASSIFICATION AUTHORITY

2b. DECLASSIFICATION / DOWNGRADING SCHEDULE

3. DISTRIBUTION / AVAILABILITY OF REPORT
This document is approved for public release; distribution is unlimited.

4. PERFORMING ORGANIZATION REPORT NUMBER(S)
ISI/SR-86-170

5. MONITORING ORGANIZATION REPORT NUMBER(S)

6a. NAME OF PERFORMING ORGANIZATION
USC/Information Sciences Institute

6b. OFFICE SYMBOL *(If applicable)*

7a. NAME OF MONITORING ORGANIZATION

6c. ADDRESS *(City, State, and ZIP Code)*
4676 Admiralty Way
Marina del Rey, CA 90292

7b. ADDRESS *(City, State, and ZIP Code)*

8a. NAME OF FUNDING / SPONSORING ORGANIZATION
Advanced Research Projects Agency

8b. OFFICE SYMBOL *(If applicable)*

9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
MDA903 81 C 0335

8c. ADDRESS *(City, State, and ZIP Code)*
1400 Wilson Boulevard
Arlington, VA 22209

10. SOURCE OF FUNDING NUMBERS

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT ACCESSION NO.
---------------	---------------	---------------	---------------

11. TITLE *(Include Security Classification)* 1985 Annual Technical Report: A Research Program in Computer Technology (Unclassified)

12. PERSONAL AUTHOR(S) ISI Research Staff

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Year, Month, Day)*	15. PAGE COUNT
Annual Technical Report	FROM July 1984 – June 1985	December 1986	156

16. SUPPLEMENTARY NOTATION

17. COSATI CODES

FIELD	GROUP	SUB-GROUP
09	02	

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*

(continued on next page)

19. ABSTRACT *(Continue on reverse if necessary and identify by block number)*

This report summarizes the research performed by USC/Information Sciences Institute from July 1, 1984, to June 30, 1985, for the Defense Advanced Research Projects Agency. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

(continued on following page)

20. DISTRIBUTION / AVAILABILITY OF ABSTRACT
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS

21. ABSTRACT SECURITY CLASSIFICATION
Unclassified

22a. NAME OF RESPONSIBLE INDIVIDUAL
Sheila Coyazo
Victor Brown

22b. TELEPHONE *(Include Area Code)*
213-822-1511

22c. OFFICE SYMBOL

DD FORM 1473, 84 MAR

83 APR edition may be used until exhausted.
All other editions are obsolete.

SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

19. KEY WORDS (continued)

1. automated implementation, Common LISP, expert system, formal specification
2. domain model, domain principles, expert systems, integration knowledge, LISP, NIKL paraphraser, optimization knowledge, program writer, XPLAIN system
3. artificial intelligence, automatic programming, consistency maintenance, formal specifications, programming environments, rapid prototyping, rule-based programming, software evolution, symbolic evalution, virtual databases
4. automation-based paradigm, GIST specification language, granularity, high-level editing, locally formal notations, mappings, reimplementations, software development, software maintenance, software specification, Will programming language
5. briefing aid application program, command graphics, computer graphics, high-level graphics language, network-based graphics, online map display
6. computer mail, gateways, interconnection, internetwork protocol, multimedia mail, networks, protocol design, protocols, protocol verification, Simple Mail Transfer Protocol (SMTP), Transmission Control Protocol (TCP), type-of-service
7. automated user interface, information dissemination, online document distribution, Strategic Computing research community
8. computer network, digital voice communication, network conferencing, packet satellite network, packet-switched networks, packet video, packet voice, secure voice transmission, signal processing, speech processing, vocoding
9. host-independent system, integrated circuit design, SUN workstation, VLSI test equipment
10. design rules, device fabrication service, device testing, integrated circuit-oriented language, MOSIS; MOS Implementation System, silicon compilation, standard pad frames, VLSI design, VLSI design library, wafer testing
11. CMOS/Bulk fabrication, MOSIS, printed circuit board fabrication, scalable design rules, VLSI
12. Janus, knowledge representation, natural language generation, Penman, Rhetorical Structure Theory (RST), text generation
13. application software, ARPANET, computer network, hardware, Interlisp, KA/LI, KL10/KL20, network services, operations, PDP11/45, QLISP, resource allocation, system software, TOPS-20, UNIX, upgrades, VAX 11/750-80, VMS
14. distributed processing, portable workstations, survivable networks
15. distributed processing, local networks, personal computers, workstation environment
16. ARPANET/MILNET access, host systems, network services support
17. computer communication networks, packet radio, survivable networks, system support

20. ABSTRACT (continued)

The ISI program consists of 17 research areas:

1. Common LISP Framework: producing an exportable version of the FSD testbed, which incorporates the well-understood portions of our research in FSD into a new automated software development paradigm, and distributing it to outside users; *2. Explainable Expert Systems:* creating a framework for building expert systems that enhances an expert system's explanatory capabilities (allowing it to explain and justify its reasoning), and eases modification and maintenance of an expert system; *3. Formalized Software Development:* studying a new automated software development paradigm in which a formal operation specification, used as a rapid prototype, is evolved to have the intended functionality, and is mechanically transformed (with human guidance) into an efficient implementation; *4. Mappings:* development of transformations for converting high-level specifications of programs into implementations in software or VLSI; *5. Command Graphics:* development of a device-independent graphics system and graphics-oriented command and control applications programs; *6. Internet Concepts Research:* exploring aspects of protocols for the interconnection of computer communication networks, specifically the design and implementation of an internetwork computer message system and the design of internetwork host and gateway protocols; *7. Strategic Computing Information System:* providing an automatic and efficient system for information exchange between Strategic Computing personnel and the research community-at-large; *8. Wideband Communication:* exploration of the technology needed to support satellite-based wideband computer communication with dynamic intercommunication of many ground stations via packet-switched sharing of a single satellite environment; *9. KITSERV:* developing cost-effective test devices for the DARPA VLSI design community, and transferring the technology to commercial companies willing to market and support the testers; *10. VLSI:* providing a low-cost, fast turnaround LSI/VLSI device fabrication service to support a geographically distributed VLSI research community with no direct access to VLSI fabrication but with access to a computer communication network, and conducting research on the VLSI design problem; *11. Advanced VLSI:* providing access to 1.2 micron CMOS/Bulk fabrication, printed circuit board fabrication, and centralized functional testing of custom-designed parts prior to their assembly into packages and boards; *12. Text Generation for Strategic Computing:* developing new methods for autonomous creation of text by machine, with the focus on fluent, easily controlled sentence and paragraph production, faithful representation of information from AI knowledge bases, and use of generated English in ongoing human-computer interaction; *13. Computer Research Support:* operation of reliable computing facilities and continuing development of advanced computing support equipment; *14. Exportable Workstation Systems:* development of a remote testbed environment of advanced workstations and servers; *15. New Computing Environment:* exploring, determining, and implementing the next generation of computers and computing facilities for the ISI research environment; *16. Strategic Computing - Development Systems:* providing development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for dissemination of the systems to the program participants; *17. Strategic C3 System Experiment Support:* participating in a Strategic Command, Control, and Communication systems experiment demonstrating and evaluating the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-based techniques).

ISI/SR-86-170

University
of Southern
California

1985
ANNUAL TECHNICAL REPORT

July 1984 - June 1985

A Research Program in Computer Technology

Principal Investigator
and Executive Director:
Keith W. Uncapher

Prepared for the Defense
Advanced Research Projects Agency
Effective date of contract 1 July 1981
Contract expiration date 31 May 1987
Contract # MDA 903 81 C 0335
ARPA Order 4242

INFORMATION
SCIENCES
INSTITUTE

213/822-1511
4676 Admiralty Way/Marina del Rey/California 90292-6695

This research is supported by the Defense Advanced Research Projects Agency under Contract No. MDA903 81 C 0335. Views and conclusions contained in this report are the authors' and should not be interpreted as representing the official opinion or policy of DARPA, the U.S. Government, or any person or agency connected with them.

CONTENTS

Summary v

Executive Overview vii

1. Common LISP Framework 1

2. Explainable Expert Systems 10

3. Formalized Software Development 20

4. Mappings 33

5. Command Graphics 43

6. Internet Concepts Research 50

7. Strategic Computing Information System 61

8. Wideband Communication 66

9. KITSERV - VLSI Kit Design Service 85

10. VLSI 89

11. Advanced VLSI 97

12. Text Generation for Strategic Computing 103

13. Computer Research Support 111

14. Exportable Workstation Systems 118

15. New Computing Environment 122

16. Strategic Computing - Development Systems 128

17. Strategic C3 System Experiment Support 132

18. Publications 135

SUMMARY

This report summarizes the research performed by USC/Information Sciences Institute from July 1, 1984, to June 30, 1985, for the Defense Advanced Research Projects Agency. The research is focused on the development of computer science and technology, which is expected to have a high DoD/military impact.

The ISI program consists of 17 research areas:

Common LISP Framework: producing an exportable version of the FSD testbed, which incorporates the well-understood portions of our research in FSD into a new automated software development paradigm, and distributing it to outside users; *Explainable Expert Systems*: creating a framework for building expert systems that enhances an expert system's explanatory capablities (allowing it to explain and justify its reasoning), and eases modification and maintenance of an expert system; *Formalized Software Development*: studying a new automated software development paradigm in which a formal operation specification, used as a rapid prototype, is evolved to have the intended functionality, and is mechanically transformed (with human guidance) into an efficient implementation; *Mappings*: development of transformations for converting high-level specifications of programs into implementations in software or VLSI; *Command Graphics*: development of a device-independent graphics system and graphics-oriented command and control applications programs; *Internet Concepts Research*: exploring aspects of protocols for the interconnections of computer communication networks, specifically the design and implementation of an internetwork computer message system and the design of internetwork host and gateway protocols; *Strategic Computing Information System*: providing an automatic and efficient system for information exchange between Strategic Computing personnel and the research community-at-large; *Wideband Communication*: exploration of the technology needed to support satellite-based wideband computer communication with dynamic intercommunication of many ground stations via packet-switched sharing of a single satellite environment; *KITSERV*: developing cost-effective test devices for the DARPA VLSI design community, and transferring the technology to commercial companies willing to market and support the testers; *VLSI*: providing a low-cost, fast turnaround LSI/VLSI device fabrication service to support a geographically distributed VLSI research community with no direct access to VLSI fabrication but with access to a computer communication network, and conducting research on the VLSI design

problem; *Advanced VLSI*: providing access to 1.2 micron CMOS/Bulk fabrication, printed circuit board fabrication, and centralized functional testing of custom-designed parts prior to their assembly into packages and boards; *Text Generation for Strategic Computing*: developing new methods for autonomous creation of text by machine, with the focus on fluent, easily controlled sentence and paragraph production, faithful representation of information from AI knowledge bases, ans use of generated English in ongoing human-computer interaction; *Computer Research Support*: operation of reliable computing facilities and continuing development of advanced support equipment; *Exportable Workstation Systems*: development of a remote testbed environment of advanced workstations and servers; *New Computing Environment*: exploring, determining, and implementing the next generation of computers and computing facilities for the ISI research environment; *Strategic Computing - Development Systems*: providing development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants; *Strategic C3 System Experiment Support*: participating in a Strategic Command, Control, and Communication systems experiment demonstrating and evaluating the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-based techniques).

EXECUTIVE OVERVIEW

As this nation becomes more dependent on information sciences, the roles and responsibilities of ISI must reflect the challenge and the opportunity.

ISI's proven flexibility in embracing relevant research and development is known. The challenge for us is to critically sense the DoD requirements well in advance of known need and provide the science, the technology, and the applications necessary to assure the military of fundamentally improved mission accomplishment.

ISI's blend of basic research systems and application expertise provides a unique advantage for DARPA and its vital support of the military.

The key areas of ISI's research and development are VLSI research and the MOSIS service, software system sciences, computer-based communication technology, and intelligent systems technology including user-friendly interfaces. In addition, we maintain a unique computer support environment for several DARPA technology programs, and support several thousand remote DoD users accessing DARPA-supported facilities at ISI.

Key to supporting DARPA transfer programs are expert systems, natural language interfaces, and fast turnaround time for custom, low-volume VLSI chips.

ISI's historic role in DARPA transfer programs represents an important coupling between the university community and many key military commands. The process is enhanced by the open nature of ISI and the availability of its research results and of the technology it generates.

ISI is often given the task of providing support efforts critical to the DARPA computer research community and to DARPA military transfer programs. We undertake these assignments with great vigor and commitment, as reflected in the details of this report.

The following summaries provide an opportunity to explore the diversity of our research and development, the systems we develop, and our involvement in DARPA transfer programs involving the military.

Common LISP Framework. The principal goal of the COMMON LISP Framework project is to support Strategic Computing (SC) contractors with a comprehensive, state-of-the-art programming framework for the development and evolution of COMMON LISP programs. This framework provides the programmer with a fileless environment of persistent objects. Their relationships with other objects are manipulated via a set of generic operations and are used to associatively retrieve the objects. The COMMON LISP Framework system (the CLF) maintains the consistency of these objects and initiates automated processing for the programmer via a set of rules. The ultimate goal of the COMMON LISP Framework project is to transfer, as it matures, this advanced technology for program specification, implementation via transformation, and reimplementation via replay of previously formalized developments on changed specifications. By incorporating successive versions of the FSD testbed into the framework SC contractors will obtain the benefits of the paradigm. It is important to emphasize that we foresee many phases in which spinoffs from the FSD testbed are incorporated into the CLF for technology transfer.

Explainable Expert Systems. The Explainable Expert Systems project, has created a framework for building expert systems that captures the knowledge necessary for providing better explanations and eases the evolution of an expert system. Our approach stems from the observation that the person who wrote an expert system can usually provide good justifications for its behavior, if he remembers the rationale behind his design decisions. Our approach is to partially replace the human system builder with an automatic programmer. We first represent, in a high-level specification language, the various kinds of knowledge that go into an expert system, such as knowledge that describes the domain, problem-solving knowledge, and terminology. The automatic programmer then uses that knowledge base to create an expert system (in LISP) from a high-level goal. As it creates the expert system, the automatic programmer records its design decisions in a development history. This development history provides the explanation routines (currently under construction) with the rationale behind the code so that, when the expert system is run, the system can explain not only how it is behaving, but also why it is behaving that way. The EES project builds on the approach first adopted in the XPLAIN system and extends it with a more advanced knowledge representation that allows us to represent additional kinds of knowledge in the high-level specification language, and a more powerful program writer that is capable of performing reformulations.

Formalized Software Development. The goal of the Formalized Software Development (FSD) project is to provide order of magnitude improvements in the cost of software development and maintenance through the use of a new software lifecycle paradigm. The new paradigm calls for the mechanical, but human-guided, derivation of software implementations from *formal specifications* of the desired software behavior. It relies on altering a system's specification and rederiving its implementation as the standard technology for software maintenance. The FSD project will demonstrate the validity of claimed benefits from the FSD paradigm in three areas: software production, software maintenance, and end-user extensibility. The project is engaged in the construction of a testbed for exploration of this new paradigm. The testbed constitutes an operational support environment for software design, implementation, and maintenance, that will support, over the project's lifetime, a succession of specification languages and implementation technologies.

Mappings. The primary goal of the Mappings project was to represent programming knowledge sources in terms of mappings between specifications and programs. These mappings fall into two broad categories: mappings for understanding the specification better and mappings for implementation of the specification. We developed a specification language, called Gist, which formalizes some of the expressive constructs commonly used in natural language specifications. These constructs include nondeterminism, descriptive reference, historical reference, constraints, and demons. A Gist specification describes the behavior of both the program and its environment; this provides a natural way to specify embedded programs that interact in complicated ways with their environment. Gist's expressive power makes it possible to specify *what* behavior a program should exhibit without saying *how* it should be implemented.

Command Graphics. The Command Graphics project builds upon previous work performed by ISI for DARPA to make graphics more readily available and to enhance decision making capability in the Military Command and Control environment. This work led to the development of an "intelligent agent" for graphics, called the Geographic Display Agent (GDA). Work in this reporting period has focused on extending the GDA's functionality. The number of available graphics display primitives was increased, a user-specifyable display precedence system was added, a transparent foreground overlay capability was developed, and a user-specifyable icon facility was incorporated. Improvements were made in the area of map generation, storage, and display. In addition, the underlying Graphics System and graphics software were ported to the SMI SUN workstation, and a new device driver for the SUN color graphics hardware was developed.

Internet Concepts Research. The long-term goals of the Internet Concepts Research project are to provide appropriate and effective designs for the primary user service applications in the internetwork communication environment. The designs are based on a set of host- and gateway-level protocols that provide the full range of service characteristics appropriate to a wide variety of applications. Our approach has been to pursue in parallel the abstract design and experimental implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions. Further, we divide our effort into several task areas: Hierarchical Naming System, Multimedia Mail, Computer Mail Interoperation, Exterior Gateway Protocol, and Studies, Surveys, and Specifications.

Strategic Computing Information System. The success of the Department of Defense's Strategic Computing research program depends on the easy dissemination of program-relatged information to a geographically distributed research community. The Strategic Computing Information System (SCIS) project provides an automatic and efficient system for information exchange between the Strategic Computing personnel and the research community-at-large. Initially, a basic SCIS was implemented on TOPS-20, using modified code origianally developed for the MOSIS VLSI brokerage service. This implementation was then used as the basis for a new, more sophisticated version that runs on a personal computer. Typical transaction of the SCIS are announcements, requests for information/information retrieval, activities related to RFPs (requests for proposals), information dissemination, queries, and administrative requests (join, address change, etc.). All transactions are handled via electronic mail.

Wideband Communication. The Wideband Communication (WBC) project, as one of several participants in the DARPA/DCA Wideband Packet Satellite Program, is involved in the development of packet voice, packet video, and high-data-rate transmissions over the Wideband Network. The primary research focus of the WBC project is to evaluate the potential of packet video. The secondary focus is to support efforts to demonstrate the feasibility of large-scale packet voice and the feasibility of other high-data-rate communication using the WBNET. The basic approach of the ISI work is to build operational systems for the WBNET which can give a practical demonstration of the potential of high-bandwidth packet-switched networks to support integrated voice, video, and data communication. Mature technology is utilized for these systems where feasible, but new technology is developed when needed. Experimentation is used to optimize the system performance for the WBNET environment. In the process, useful information can be developed for tuning the Wideband Network itself to the needs of voice, video, and data communication.

KITSERV VLSI Kit Design Service. The goal of the KITSERV project is to develop cost-effective test capability for VLSI designers as an integral part of their design environments, and to make this capability available to a large population of designers by transferring the technology to commercial companies willing to market and support the testers. KITSERV will design and prototype a family of test environments to fill the designers' need to verify designs. These testers will cover a wide spectrum of capabilities, from simple stand-alone testers to complex systems operating in conjunction with VLSI design tools running on host machines.

VLSI. The VLSI design communities of DARPA and NSF require fabrication capabilities in order to investigate the design methodologies and architectures appropriate to VLSI where gate-count will exceed one million gates per device. Recognizing this need, DARPA established MOSIS (MOS Implementation Service) at ISI in January of 1981. MOSIS has reduced the cost of VLSI prototyping, shortened turnaround time for VLSI prototyping, freed designers from fabrication idiosyncrasies, and made design less dependent on specific fabrication lines. MOSIS is developing a library of community designs and making it available to new designers. This will allow the community to build on the work already done and will help designers avoid "reinventing the wheel." In addition to this library, MOSIS has acquired from DARPA a library-designed standard cell library for 3 micron P-well CMOS/Bulk technology, designed in rules identical to those of MOSIS and therefore capable of being fabricated by MOSIS vendors. MOSIS is making this library available to its user community. During the reporting period MOSIS nearly completed the transition from KL-2060s running TOPS-20 to a cluster of VAX-11/750s. MOSIS substantially expanded the vendor base. In addition, MOSIS developed partial functional testing of wafers and a set of standard pad frames to reduce packaging costs.

Advanced VLSI. The primary goals of the Advanced VLSI project are to provide the DARPA VLSI systems design community access to 1.2 micron CMOS/Bulk technology and to printed circuit board fabrication; and to provide centralized functional testing of custom-designed parts prior to their assembly into packages and boards.

Text Generation for Strategic Computing. This project is creating and demonstrating new technology to provide an English-in, English-out interface to computer data, called Janus. ISI is developing the English-out (text generation) portion of Janus, referred to as Penman. Initial capability will be demonstrated on a naval database, but most of the techniques are more general and will be able to be reapplied to other military problems. The end result will be an exciting new capability for the Military that produces answers to queries and commands in the form of text that is understandable to any user who understands English.

Computer Research Support. The Computer Research Support project is responsible for providing reliable computing facilities on a 24-hour, 7-day schedule to the ARPANET/MILNET research and development community. The project also makes available to ARPANET/MILNET users the latest upgrades and revisions of hardware and software. The project provides continuous computer center supervision and operation, and a full-time customer-service staff that is responsive to user inquiries. This project supports two computer installations, at ISI's main facility in Marina del Rey and at the Design Center at Gunter Air Force Base in Alabama.

The ISI Information Processing Center provides support in four tightly interrelated, though distinct, areas: Hardware, System Software, Operations, and Network Services. Each group is very concerned about both the effective use of the machines and the security of the physical plant and the files and other online information.

Exportable Workstation Systems. The primary goal of this project is to install and support a workstation environment at the DARPA-ISTO offices in Arlington, Virginia. It is also intended to provide a testbed for several other concepts and facilities. The environment is configured to accommodate a variety of workstation models with varying capabilities.

New Computing Environment. The New Computing Environment (NCE) project's goal is to adapt developing computer technologies to serve the research and military user communities. The resulting model computing environment will serve several purposes. It will provide a very large improvement in languages, system support, and additional computing cycles to the ongoing ISI research effort; it will serve as a testbed and eventual existence proof for the implemented technology; and it will serve as a proving ground for local computer environments targeted for DARPA and the military community, as well as a device for investigating new research ideas that will eventually be adapted by those communities. In addition, the small size, portability, and local computing capability of personal computers will allow for experimentation and realization of command and control requirements for an environment that can exist in mobile command centers.

Strategic Computing - Development Systems. This project provides development computers for the DARPA Strategic Computing program, system integration as required, and distribution mechanisms for disseminating the systems to the program participants. In addition, the project is defining an architecture for system communications and resource sharing among the computers acquired.

Strategic C3 System Experiment Support. DARPA has defined an experiment in Strategic C3 systems to be conducted in cooperation with the World Wide Military Command Control System (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC). The concept of the experiment is to demonstrate and evaluate the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-base techniques) for strategic command, control, and communication. ISI is providing an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with this technology and to ensure the success of the experiment. Specifically, SAC must have fairly broad-based experience with ARPANET-based online interactive computing. Installation and maintenance of modems and of 55 or more interactive CRT terminals, user training, system software training and support, and on-site maintenance of equipment for the 300+ users at SAC Headquarters at Offutt Field, Omaha, Nebraska and at a handful of distributed SAC bases, are part of the continuing program.

1. COMMON LISP FRAMEWORK

Research Staff:	**Support Staff:**
Robert M. Balzer David S. Wile Dennis Allard Chloe Holg Donald Voreck William Vrotney	Audree Beal

1.1 PROBLEM BEING SOLVED

The success of many components of the Strategic Computing program relies heavily on knowledge-based systems, systems in which some aspect of human knowledge about a problem area has been captured in rules for reasoning about and solving the problem. A number of commercial firms that specialize in providing such "expert system" tools have arisen and many practical systems with military significance have been developed: for example, systems for equipment diagnosis, signal interpretation, and battle management. There are two approaches to improving services developed using expert system technologies. The first attempts to improve the language that expresses the knowledge used in the system and simultaneously to develop automatic tools to improve the performance and understandability of such systems.[1] The second, complimentary approach suggests substitution of task-specific, hand-tailored software for some of the knowledge sources in a system, where some problem feature makes their encoding as "rules" inappropriate or inefficient.

Although both approaches are important, this latter problem is especially important to ensuring the success of the SC program, wherein specialized equipment may have to be modelled with more fidelity than in more mundane expert system tasks. The COMMON LISP Framework project originally arose to support exactly this military problem: to facilitate SC contractors' development of problem-specific software in an expert system context. Almost every currently available knowledge-based system is programmed in some dialect of LISP. Hence, the SC program has procured LISP-based workstations for development of expert systems to effect the three primary military tasks it intends to accomplish. In order to maintain independence from the particular hardware environment on which the expert systems are developed, a standard dialect, called COMMON LISP, has been designed and will be required to run on each of these supporting machines. This language will be used to capture the task-specific expertise used in conjunction with more general rules in a knowledge base. Although the use of the COMMON LISP language unifies expression of the knowledge in such systems, the diversity of hardware support environments makes the assumptions underlying the frameworks of such systems highly important. Code cannot be moved between hardware configurations unless it relies on the same underlying facilities, as well as the same language. The COMMON LISP Framework project provides a uniform framework across many vendors' hardware, in which SC contractors can develop and evolve their systems.

---

[1] Such an approach is being pursued by the Explainable Expert Systems project at ISI.

1.2 GOALS AND APPROACH

1.2.1 Goal: State-of-the-Art Framework for Common LISP Program Development

The principal goal of the COMMON LISP Framework project is to support Strategic Computing (SC) contractors with a comprehensive, state-of-the-art programming framework for the development and evolution of COMMON LISP programs. This framework provides the programmer with a fileless environment of persistent objects. Their relationships with other objects are manipulated via a set of generic operations and are used to associatively retrieve the objects. The COMMON LISP Framework system (the CLF) maintains the consistency of these objects and initiates automated processing for the programmer via a set of rules.

For several years ISI's Software Sciences Division has been developing the technology to support a new software lifecycle paradigm. This paradigm calls for the mechanical, but human-guided, derivation of software implementations from *formal specifications* of the desired software behavior. It relies on altering a system's specification and rederiving its implementation as the standard technology for software maintenance. Practical use of this paradigm requires extending active machine involvement into the earliest stages of the development process. The Formalized Software Development Project was established at ISI to study the feasibility of this approach on realistic sized programs. It has developed a testbed system in which these ideas of specification and automated implementation may be studied through hands-on experimentation.

The ultimate goal of the COMMON LISP Framework project is to transfer, as it matures, this advanced technology for program specification, implementation via transformation, and reimplementation via replay of previously formalized developments on changed specifications. By incorporating successive versions of the FSD testbed into the framework SC contractors will obtain the benefits of the paradigm. It is important to emphasize that we foresee many phases in which spinoffs from the FSD testbed are incorporated into the CLF for technology transfer.

1.2.2 Approach: Technology Transfer from the FSD Testbed

The idea for the COMMON LISP Framework project arose as DARPA recognized the importance of the ongoing research in the Formalized System Development project at ISI, especially its potential for use by its SC contractors. The Formalized System Development Testbed System (see Chapter 3)--hereafter called the testbed--is being developed to support research in domain modelling, expert systems, programming methodology, and administrative service development. Its major strengths are as follows:

- It provides a uniform, persistent, fully associative database to access all information. This constitutes a finer grain size than a file system and is the basis for integrating mundane, daily activities (mail, text editing, calendars, etc.) with the programming service.
- Within the database, the user can provide monitoring programs to maintain the integrity of his data automatically, and can provide rules which react demonically to changes in the state of the database to automatically accomplish activity which presently requires manual decision making and action.
- A consistent, model-based framework for interaction with the database is provided to unify generic activities such as editing, viewing, scrolling, deleting, and focussing.
- Program construction is done via the cycle of specification, examination of behavior, and transformation to implementation, followed by respecification, reexamination of behavior, and retransformation using the previously recorded development history.

Three features distinguish the testbed from other programming environments:

- It supports associative retrieval in a persistent object-base whose consistency is automatically maintained by rules.
- It actively, automatically supports mundane programmer activity via demonic rule invocation based on object-base changes.
- It supports program evolution and maintenance via recording and structuring a "development history" of the programming process.

The system is being used to develop itself, forcing it to mature in realistic directions. Although research problems continue to abound in the development of this system, this maturity made it ripe for technology transfer through this effort.

The FSD testbed is designed to support all aspects of computing, from the mundane activities of mail handling and document preparation, through programming, to advanced software methodology experimentation. The major role of the COMMON LISP Framework project was initially to extract that portion of the environment that will be universally useful to the SC community. Identifying that subset and making it robust enough for real use constituted the major problem for the COMMON LISP Framework project in its first year. Although the project only began late in 1984, a preliminary version of the CLF is expected to be in beta test by the end of 1985 and to be delivered to other SC contractors in early 1986.

1.3 SCIENTIFIC PROGRESS

From the outset, the COMMON LISP Framework was intended to go through three major phases. The functionality of the initial CLF can be best characterized as flexible management of program objects and tools. A preliminary version of this portion is in alpha test. It should be in beta test at the end of 1985 and delivered to other SC contractors in early 1986. During the second phase, the tools provided in the initial system will be made more reliable and robust. The third phase will begin to incorporate the FSD technology for program specification, implementation via transformation, and reimplementation via replay of previously formalized developments on changed specifications.

1.3.1 Salient Features of the CLF

The COMMON LISP Framework comprises the following major components:

- AI Operating System (CLF Kernel);
- Program Management Service;
- Program Development Service.

The AI operating system kernel of the CLF provides the programmer with a fileless environment of persistent objects. Their relationships with other objects are manipulated via a set of generic operations and are used to associatively retrieve the objects. The CLF maintains the consistency of these objects and initiates automated processing for the programmer via a set of rules.

A unique feature of the CLF is that all objects manipulated by programmers are represented in this persistent object-base--specifically structured objects, like modules with components, as well as primitive program structures, like function, variable, and structure declarations. Also, relationships and objects elsewhere represented in a very ad hoc, diversified fashion, such as flow analysis relationships, versions, time stamps, developers, and users, are all represented uniformly in the object-base.

The kernel provides generic facilities for manipulating these programming objects as objects. In addition, special facilities have been introduced into the framework to provide programming assistance in the following areas:

- Module creation;
- Component addition;
- Importing existing files;
- Component modification and editing;
- Code installation.

Each of these facilities requires user intervention or initiation.

Especially important are the facilities for managing the consistency of program information, built using the rule-based kernel. These facilities perform the following activities:

- Automatically compile functions that are "installed" by the user;
- Allow multiple-buffer editing of the same object while maintaining correct views in each buffer;
- Automatically (re)analyze functions when they are installed;
- Automatically maintain program object ordering by load-order and view-order.

Each of these activities is managed differently in existing programming environments. Some are managed by the user only--the system provides no help for such consistency maintenance.

The CLF's Program Development Service provides automated maintenance documentation by maintaining an annotated development history. The user is responsible for providing development step annotations briefly characterizing his activity when he changes the program. The programmer may explicitly indicate substructure in his development, whereupon the system maintains his stated goal structure.

Of considerable importance is the ability of a user to indicate his plans for future programming activity, through creation of development steps called "pending steps." The user can subsequently--perhaps at a much later time--handle these development steps; the system automatically incorporates them as new steps in the existing structured history.

Not only is the development service able to recall the history of the programming activity itself, but it is linked into the persistent object-base management activity in such a way that information associated with the changes may be used for maintenance documentation and release and for version management. Thus, since the generic object-base facility is used to store the development history *itself*, one can find all the development steps affecting a particular object, as well as all the objects affected by a particular development step.

This link into the incremental saving mechanism allows the development service to provide automated distribution of program objects to users of the systems of which they are components, as well as to aid in tracking the installation of sets of changed program objects when the affecting development steps are accepted. The documentation used to describe development steps is used to document the distributed changes to users of the system. This is a particularly interesting side-effect of our efforts to record as much as possible of the programming *process* in the machine, where the information can be analyzed and the user aided based on this analysis.

1.3.2 Accomplishments

The major accomplishments of the CLF project include:

- Identification of the relevant subset of FSD facilities to be released in a "beta test" version of the CLF in December.
- Began alpha test of:
    - The relational database of program objects (modules, functions, record declarations, etc.), analysis predicates (flow relationships, call relationships, variable usage patterns) via which all tools communicate.
    - The testbed facilities for creation, modification, destruction, retention, organization, and retrieval of these objects.
    - The interactive standardized user interface to these facilities using a menu/window system.
    - Tools which interface to these objects include the ZMACS editor and a batch invoked tool for program- and data-flow analysis.

  - o The "DEVELOP" mechanism to record a history of change within the FSD environment.

- A facility for converting existing programs for use in the framework will be available.
- A facility for managing version control and system releases will be available on top of the DEVELOP facility mentioned above.

1.4 IMPACT

The COMMON LISP Framework project directly supports the Strategic Computing program in a very fundamental way. We emphasized earlier the extent to which the COMMON LISP Framework provides a uniform base for development of expert systems in the various programs funded by the Strategic Computing program and how the COMMON LISP Framework can enhance expert systems' capabilities by providing for substitution of task-specific, hand-tailored software for some of the knowledge sources in a system, where some problem feature makes their encoding as "rules" inappropriate or inefficient.

In fact, the COMMON LISP Framework has a secondary benefit to the military, which ultimately could be more important than its original purpose. The COMMON LISP Framework project provides an advanced, object-oriented programming environment for the development of COMMON LISP programs and a framework of conventions, structures, and models into which users can integrate their software. Such a powerful programming environment has **never** been available to anyone, let alone the military. Its availability to military programmers should improve their output, and decrease the time and money needed for maintenance of programs written for military purposes. In addition, the framework will be used as a model environment for more compilation-oriented languages, like ADA, already proven to be of military benefit.

1.5 FUTURE WORK

CLF is basically a technology transfer project, adapting well-understood aspects of the FSD system to COMMON LISP programming support. The development cycle of functionality obeys the diagram in Figure 1-1. As each system feature matures in the FSD system, it becomes incorporated into the CLF. That CLF facility then becomes the foundation for the next version of the FSD system. Generally, features undergo two distinct phases. The initial "demonstration" phase (FSD Demo_n) is incomplete with respect to some intended functionality, but demonstrates a portion of that intended

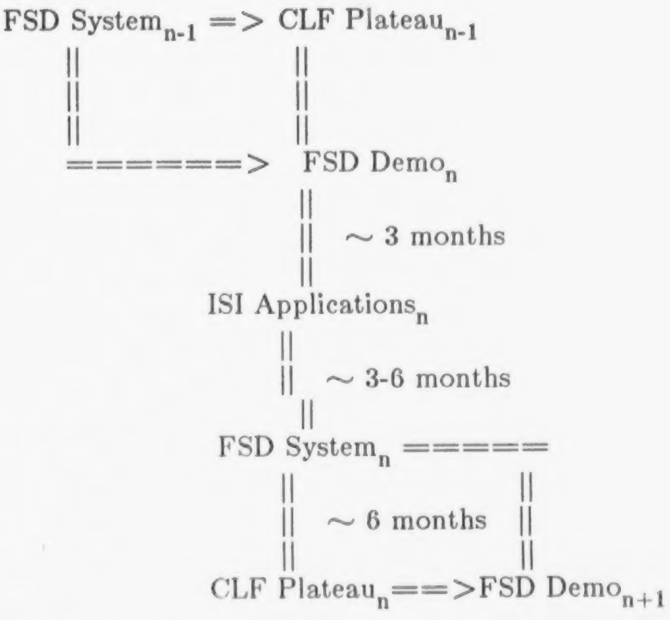

Figure 1-1: FSD-CLF plateau cycle

functionality as a straw man--a rapid prototype. After about three months, the demonstrated features are robust enough to endure "alpha test" in ISI applications. They then become official FSD system features (FSD System$_n$) and are "beta tested" at ISI for several months by a wider user community before being incorporated into the CLF (CLF Plateau$_n$). This plateau then becomes the base of the next FSD demonstration.

1.5.1 Planned Program

An initial Common LISP Framework will be available (ready for "beta test") by the end of 1985. Although the functionality to be delivered in the initial CLF provides flexible data management of all program objects and tools, we do not expect this version of the CLF to run efficiently enough for large program development. Hence, the next year will principally be spent improving the facilities provided in the "beta test" version of the system and in allowing programmer access to some of the foundational facilities used by the CLF. The improvements will involve both performance and ergonomic issues. Access to the underlying CLF facilities will be necessary so that the CLF users can provide a "consistent underlying viewpoint" to the users of their facilities.

Within the next year, there will be more user access to functionality underlying the delivered system:

- A BNF grammar-based parsing mechanism where more sophisticated structure is desired.
- Programmatic interface to the object database, via a released package of functions for object manipulation.
- Programmatic interface to the object-saving mechanism to maintain persistent user-defined objects.

We will also begin to keep track of activity histories, allowing undo, redo, etc. In addition, FSD representation of the run-time stack will permit definition of enhanced debugging facilities.

By the end of the contract year, the beta-test CLF will be hand-derived from the FSD testbed. We will build primitive automatic mechanisms for extraction of this functionality from the FSD Testbed system on top of the FSD DEVELOP system. It will consist mostly of subsetting facilities (a natural activity with the associative database descriptive facilities), used to eliminate testbed functionality dealing with personal database maintenance and higher level program maintenance.

2. EXPLAINABLE EXPERT SYSTEMS

Research Staff:	**Research Assistants:**	**Support Staff:**
William R. Swartout	Johanna Moore	Audree Beal
Robert Neches	Jody Paul	

2.1 PROBLEM BEING SOLVED

Even the best expert system is worthless if it is not accepted by its intended users. A critical factor for user acceptance of expert systems is that they must be able to explain their reasoning. Recent surveys of potential users of expert systems (for example, [9]) have shown that, among the capabilities surveyed, explanation was regarded as being essential--more important than automatic knowledge acquisition, common-sense reasoning, ease of use, or even being error-free. An explanation facility can:

- give the user confidence in the system's results,
- warn him if the system is being applied inappropriately,
- aid system developers in debugging/developing the system, and
- help to educate the user.

The development of an expert system is an evolutionary process. We usually lack sufficient foresight to be able to precisely specify what a system should do prior to constructing it. We form a better understanding of both the problem and the requirements for the system through experimentation with a prototype that we modify and extend until it performs adequately. Due to this experiential, incremental nature of development, it is critical that an expert system shell support the addition, modification, and deletion of knowledge with minimal perturbation of the rest of the knowledge base, and that a developer be able to determine how a proposed change will affect the behavior of the system.

EES is an important step in easing the process of creating and maintaining expert systems. EES provides leverage in three important ways. First, EES's explanatory capabilities can be used by a maintainer to better understand an expert system's current capabilities and the effects of proposed changes. Second, expert systems created using EES are inherently more modular than those created using conventional expert system shells. This makes it easier to change one part of the system independently of the rest (and opens up the possibility of knowledge reuse). Third, the EES approach will provide a permanent record of major design decisions by the designer and of subsequent decisions by others improving a particular expert system developed under EES. With the frequent turnover of computer personnel, such a record provides a valuable "corporate history" of the development of an expert system.

Despite its importance, an explanation facility is frequently regarded as an add-on, something that can be retrofitted to an expert system once the basic system is complete. This attitude is unfortunate, because it severely limits the quality and range of explanations that can be offered. It takes more knowledge to solve a problem and explain the solution than it does to just solve the problem. Even when explanations are provided, they are largely limited to cookbook-like paraphrases of what the system does or did. A number of important kinds of explanations, such as justifications, cannot be supported because the knowledge needed to provide them is not represented [1, 6]. Additionally, once expert systems grow to realistic size it becomes quite difficult to modify and evolve them. We feel these problems stem from some fundamental flaws in the underlying expert system architecture:

- **No separation of concerns.** Current frameworks do not support a modular representation of the knowledge in an expert system. The rules or methods incorporate many different kinds of knowledge, such as domain facts, heuristics for achieving goals, efficiency concerns, and so forth, that should be represented separately and explicitly but never are. As a result, the explanations such systems can provide are often opaque. This intertwining of knowledge also reduces a system's modularity and makes it more difficult to modify.
- **No record of operationalization of abstract concerns.** The process of compiling general domain facts and problem-solving knowledge into specific rules is performed, unrecorded, in the system builder's head. The design decisions behind the system are not available; hence, machine-produced justifications of its behavior cannot be provided. This also makes the system more difficult to modify or evolve, because the system builder is forced to work at a low level of representation, mentally compiling high-level changes into low-level rules.
- **Limited explanation techniques.** Even when a framework does provide an explanation facility, the facility usually suffers from some serious limitations. In particular, such facilities are inflexible, and usually provide only one way of presenting an explanation. They are also difficult to modify: explanation methods are currently hand-coded in LISP. In addition, they are not responsive to the user's needs: if the user doesn't understand the explanation, there is no way for him to indicate to the system exactly what it is that he doesn't understand.

To summarize, problems in explanation, maintenance, and extension can be traced to the same underlying causes: the lack of a record of the system's development, and the use of a low-level, relatively compiled representation for rules or methods.

2.2 GOALS AND APPROACH

In the Explainable Expert Systems project, we have created a framework for building expert systems that captures the knowledge necessary for providing better explanations and eases the evolution of an expert system. Our approach stems from the observation that the person who wrote an expert system can usually provide good justifications for its behavior, if he remembers the rationale behind his design decisions. Our approach is to partially replace the human system builder with an automatic programmer. We first represent, in a high-level specification language, the various kinds of knowledge that go into an expert system, such as knowledge that describes the domain, problem-solving knowledge, and terminology. The automatic programmer then uses that knowledge base to create an expert system (in LISP) from a high-level goal. As it creates the expert system, the automatic programmer records its design decisions in a development history. This development history provides the explanation routines (currently under construction) with the rationale behind the code so that, when the expert system is run, the system can explain not only how it is behaving, but also why it is behaving that way. The EES project builds on the approach first adopted in the XPLAIN system [6] and extends it with a more advanced knowledge representation that allows us to represent additional kinds of knowledge in the high-level specification language, and a more powerful program writer that is capable of performing reformulations.

2.3 SCIENTIFIC PROGRESS

In developing EES, we began by determining what kinds of explanations expert systems needed to offer. Using protocols and our own experience as expert system builders and users, we identified approximately a dozen different classes of useful explanations (see [7]). Based on that investigation, we identified different kinds of knowledge needed to support such explanations. We have provided representations for several of them, including domain-descriptive knowledge, problem-solving knowledge, and terminology [4, 3]. EES uses NIKL [2] as its underlying knowledge base. This provides one of the key advances of EES over XPLAIN: the explicit separation of terminology. We have identified this as a critical factor in enhancing the evolvability and explainability of expert systems [8].

The EES program writer, currently under development, is the central component of the EES system. Using a high-level specification for an expert system, the program writer will produce a LISP implementation for the expert system. The program writer will use two passes. During the first pass, the basic structure of the expert system is designed, and that design is recorded in a machine-readable development history. The second pass of the system examines the recorded design and produces the actual LISP implementation. The first pass is substantially more complex and will be tested on two domains: a software tool, called the *Program Enhancement Advisor*, and a diagnostic system for telemetry systems. The second pass of the program writer will be tested on the second domain and is intended to produce a demonstration expert system that is approximately 10 pages of LISP code in length.

The program writer primarily creates the implementation by refining high-level goals into increasingly lower level goals until the level of system primitives is reached. The major advance of the EES program writer over XPLAIN's program writer [6] is that the EES program writer can reformulate a goal into other goals if no plan can be found for implementing the goal. The program writer makes use of several kinds of reformulations that seem to occur frequently in expert systems (see [4]). Providing for reformulation has had several benefits. First, it allows us to more accurately model the process of creating an expert system, because the problem of reformulating desired goals in light of available implementation technology is in fact a major part of expert system design. Second, by further relaxing the coupling between plans and goals, this approach should facilitate knowledge reuse across expert systems.

2.4 IMPACT

We expect the EES project to provide a new paradigm for developing expert systems that will:

- provide better explanations, because the rationale behind the system will be captured;
- be easier to extend and modify, because they will be more modular;
- be more sound, because their development will be more principled.

2.5 FUTURE WORK

The EES program writer should become operational in the next contract year. As stated above, the program writer will use a high-level specification to produce a LISP implementation for an expert system. In addition, we will extend the EES framework in three areas:

- **Specification.** As mentioned above, we have identified several kinds of explanations that expert systems should be able to provide. To provide some of these, it will be necessary to capture additional kinds of knowledge in the underlying specification language that EES provides for constructing expert systems. This will allow us to provide these explanations and, by increasing the modularity of the expert systems, make them more maintainable.
- **Operationalization.** We feel that the fact that EES explicitly provides both a specification and an implementation for an expert system, and a development history that connects the two, will allow us to mimic human expert problem-solving behavior more closely than can be done with conventional expert systems through the use of "compiled-in assumptions," which we describe below.
- **Explanation.** We want to provide a flexible explanation facility that can fully exploit the knowledge represented within the EES framework. Further, we want this explanation facility to be capable of reacting to user misunderstandings and presenting additional clarifying explanations.

An additional activity involves use of the framework:

- **Support for the expert system lifecycle.** We want to construct knowledge acquistion aids that exploit the framework to provide *support for the expert system lifecycle*, and we want to test this entire approach through *real use* in the construction of actual expert systems.

In the remainder of this section we discuss our plans in greater detail.

2.5.1 Specification

There are several additional kinds of knowledge we would like to be able to represent in our framework in order to offer better explanations and be better able to capture the process of building an expert system.

- **Preference/cost-benefit knowledge.** Cost-benefit knowledge is associated with a plan to indicate the costs or benefits involved in adopting it (e.g., fast execution versus efficient use of storage). Preference knowledge is associated with goals to induce an ordering on candidate plans based on their cost-benefits (e.g., prefer methods that are space-efficient). Representing preference/cost-benefit knowledge is necessary to be able to explain why one strategy was preferred over another, even when both could, in principle, achieve the same goal.
- **Knowledge of problem-solving methods.** A *method* consists of a sequence of *actions* which may be engaged to effect the solution of a problem. Those actions may involve specific problem-solving operators, such as actions which may be used to generate or test a candidate for solution; or they may involve the creation of other problems whose results, if successfully solved, may be drawn upon to solve the original problem. This effort would

be directed toward developing a schemata-based representation of general methods. This representation would allow us to represent knowledge about so-called "weak" methods and represent their link to strong methods. Currently, EES only represents problem-solving knowledge in domain-specific terms (e.g., "to diagnose a component, consider its subcomponents"). We have found such a representation to be inadequate for representating integration and optimization knowledge. This effort would provide another way of describing problem-solving knowledge in methodological terms (e.g., "consider diagnosis as a tree search problem").

- **Integration knowledge.** A method which involves the creation of other problems must also include knowledge of what to do with the results of succeeding or failing to solve those problems. Such knowledge will be called *integration* knowledge. Representation of this knowledge must accommodate several elements:

    - *history knowledge* of those problems which have already been posed, which were solved, which could not be solved, and what information was provided by those problems which were solved.
    - *control knowledge* of whether or not more problems need to be considered and, if so, which ones.
    - *combination knowledge* of how the solution to the original problem may be synthesized from the solutions of the problems which have been constructed and from the knowledge of those problems which could not be solved.

    The need for integration arises frequently in expert systems (as, for example, when multiple rules may assert differing levels of belief in the same hypothesis). In conventional expert system frameworks, this integration problem is often dealt with by some mechanism built into the rule interpreter (e.g., MYCIN's certainty factor mechanism). That approach has several problems:

    - The mechanism for integration cannot be explained, since it is built into the framework directly.
    - The assumptions underlying the mechanism are not checked, so abuse is possible.
    - Usually, only one integration mechanism is available; it is not possible to choose among several.

    We intend to ameliorate these problems by explicitly representing integration knowledge and having the program writer reason about this knowledge to select the most appropriate integration mechanism(s) when the expert system is created.

- **Optimization knowledge.** This is the knowledge that makes the expert system efficient. It includes knowledge about how to collapse goals and optimize scheduling. By representing this knowledge (and its application) explicitly, we will be able to answer questions such as why a particular test was scheduled in a particular way.

Finally, we would like to construct a **reusable top-level knowledge base**. Almost all current knowledge representation formalisms require the user to construct a knowledge base from scratch. That is, the user is only provided with some top-level concept (e.g., "THING") and a set of primitive operations for constructing new concepts. This imposes a significant burden on the system builder, because he must create the entire top level of the concept hierarchy in addition to representing the particular domain knowledge he is interested in. Creating the top level is hard, because the system builder is confronted by many seemingly arbitrary decisions; but it is also critical, because it provides the basis for the entire knowledge base. We intend to relieve the system builder of this task by creating a reusable top-level knowledge base. We will use natural language and the distinctions forced by explanation as a guide in structuring the knowledge base. This will provide several benefits:

- **Easier system construction.** Knowledge for a new expert system would be provided by *specializing* this existing knowledge base rather than constructing everything from scratch.
- **Greater potential for knowledge reuse.** Since systems would be constructed starting from the same underlying base, there would be greater opportunities for reusing knowledge across systems than would be possible if each system were constructed from a different (and arbitrarily structured) knowledge base.
- **Guidance for style.** The predefined top-level knowledge base could acquaint a user with the appropriate style for using the knowledge representation.

2.5.2 Operationalization

We would like to address two areas in improving operationalization:

- **Interpretation.** Currently, the EES framework compiles the specification for the expert system into an implementation. We would like to modify the framework so that it could support direct interpretation of the specification as well as compilation. This would allow an expert system designer to more rapidly experiment with design changes, and also would allow us to make use of "compiled-in assumptions," described below.
- **Compiled-in assumptions.** It is well known that experts make assumptions. Sometimes an expert will explicitly make an assumption during problem-solving, but often assumptions are compiled into a problem-solving method so that, by adopting a particular method, the expert implicitly makes assumptions. For example, although a novice might methodically consider and reject very unlikely problems when diagnosing a device, a more experienced troubleshooter won't consider them, assuming they aren't causing the problem because they are so improbable. He doesn't even appear to be consciously aware that he is making these assumptions. An intriguing possibility for EES would be for the program writer to compile

such assumptions into the code it writes. This would make the program writer a non-equivalence-preserving compiler, since the deep specification knowledge would be assumption-free, but the compiled knowledge would contain assumptions. These assumptions would be explicitly recorded in the refinement structure so that they could be retracted later if they appeared to be wrong.

Making such assumptions could significantly improve the efficiency of an expert system in terms of both runtime and the amount of data for which it queried the user. There seem to be three major considerations in making an assumption:

1. it must be correct most of the time,
2. it must be possible to determine when it is violated, and
3. it must be possible to recover from an assumption violation.

The benefit of this approach is that it would allow us to create expert systems that solve straightforward cases quickly, while still being robust enough to handle complex cases.

2.5.3 Explanations

To provide better explanations, explanation will be treated as a planning problem, where the system plans an explanation to provide the user with the knowledge he appears to need to know. This approach is:

- Flexible: because the system can represent many different explanation strategies and employ them as circumstances warrant.
- Easy to modify: strategies provide a more modular organization.
- Reactive to the user: if the user appears not to understand the explanation, the system will either replan a new explanation or interactively debug the current explanation.

2.5.4 Support for Expert System Lifecycle

In the approach to expert system construction that we are investigating, more knowledge must be represented than would be required in conventional approaches. However, because a richer and more principled knowledge base is constructed, it should be possible to provide enhanced tools for maintaining and evolving expert systems. We intend to create aids both for initial system construction and for system evolution.

2.5.4.1 Aid for initial system construction

The reusable top-level knowledge base (mentioned above) can aid a system builder in constructing an initial version of a system by laying out much of the top structure of the knowledge base that he would otherwise have to create himself. Results:

- The expert system is constructed more quickly.
- The system builder avoids having to make many arbitrary decisions about the top level.
- Greater possibilities exist for knowledge reuse across systems, since system builders start from a common base.
- The system builder is given guidance with a "style" of representation.

2.5.4.2 Support for system evolution

Because expert systems are constructed in an incremental fashion, they must be easily evolved and enhanced. Our approach to expert system construction inherently provides some benefits for evolution, such as modification at the specification level and explanation. In addition, we propose to construct a knowledge acquisition aid that will take advantage of the rich structure of an EES knowledge base.

- *Modification occurs at the specification level, not the implementation level.* The expert system is easier to modify since it is more modular. This capability is a natural result of our approach to expert system development.
- *Explanation.* The explanation facility provided by EES can be used by system developers (as well as end users) to better understand the knowledge base. By presenting knowledge from a different point of view (e.g., natural language as opposed to formal notation), an explanation facility can often make errors glaringly apparent [5]. The system should also be able to describe the effects of a proposed change in the knowledge base.
- *Principled knowledge acquisition.* The need to construct a more explicit knowledge base is a potential problem for EES users, but at the same time it provides an opportunity to provide powerful acquisition aids which address that problem. To aid in constructing the detailed NIKL models that our approach requires, we will develop a knowledge acquisition aid. This system will help users construct and augment models. It will employ knowledge of how NIKL models are constructed, possible pitfalls, and detailed knowledge of how various parts of a model interact, to guide the user in constructing a model. When an existing expert system is being extended, the acquisition system will use the rich EES knowledge base to discover how the system was constructed and what goals it is intended to achieve, to guide and constrain the acquisition of further knowledge. An important aspect of this system is that it will be user extendable, so that users can get assistance with domain-specific as well as domain-independent modelling issues.

REFERENCES

1. Clancey, W., "The Epistemology of a Rule-Based Expert System: A Framework for Explanation," *Artificial Intelligence* 20, (3), 1983, 215-251.

2. Moser, M. G., "An Overview of NIKL, the New Implementation of KL-ONE," in *Research in Natural Language Understanding*, Bolt Beranek and Newman, Inc., Cambridge, Mass., 1983. BBN Technical Report 5421.

3. Mostow, J., and W. Swartout, "Towards explicit integration of knowledge in expert systems: An analysis of MYCIN's therapy selection Algorithm," in *Proceedings of the National Conference on Artificial Intelligence*, 1986. (Accepted for publication.)

4. Neches, R., W. Swartout, and J. Moore, "Enhanced maintenance and explanation of expert systems through explicit models of their development," *Transactions On Software Engineering*, November 1985. Revised version of article in *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, December 1984.

5. Swartout, W., "GIST English generator," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, 1982.

6. Swartout, W., "XPLAIN: A system for creating and explaining expert consulting systems," *Artificial Intelligence* 21, (3), September 1983, 285-325. Also available as USC/Information Sciences Institute, RS-83-4.

7. Swartout, W., "Knowledge needed for expert system explanation," *Future Computing Systems*, 1986. (Revised version of paper in NCC '85, accepted for publication.)

8. Swartout, W., and R. Neches, "The shifting terminological space: An impediment to evolvability," in *Proceedings of the National Conference on Artificial Intelligence*, 1986. (Accepted for publication.)

9. Teach, R. L., and E. H. Shortliffe, "An analysis of physicians' attitudes," in B. G. Buchanan and E. H. Shortliffe (eds.), *Rule-Based Expert Systems: The MYCIN Experiments of the Stanford Heuristic Programming Project*, Addison-Wesley, Reading, Mass., 1984.

3. FORMALIZED SOFTWARE DEVELOPMENT

Research Staff:	**Support Staff:**
Neil Goldman	Audree Beal
Tom Kaczmarek	Hillel Gazit
Don Cohen	Seth Goldman
Michael Fox	Edward Ipser
Lewis Johnson	Mark Sausville
Kirk Kandt	Jay Skeer
Robert Neches	

3.1 PROBLEM BEING SOLVED

The goal of the Formalized Software Development (FSD) project is to provide order of magnitude improvements in the cost of software development and maintenance through the use of a new software lifecycle paradigm. The new paradigm calls for the mechanical, but human-guided, derivation of software implementations from *formal specifications* of the desired software behavior. It relies on altering a system's specification and rederiving its implementation as the standard technology for software maintenance [1].

The specification-based software paridigm enables system builders to separate concerns, methodologically and notationally, that are not separable in the paradigm prevalent today. Specifically, issues of *functional behavior* of a system can be expressed and explored without the necessity for considering any specific *implementations* that produce the behavior. Implementation concerns can be expressed, and alternative implementation strategies explored, in the context of a fixed specification. This separation of concerns is possible today only with the use of *informal* specifications. Informal specifications, however, are susceptible to multiple interpretations, to misinterpretation, and to errors of omission. But most significant is the fact that they are not interpretable *by machine* at all. This prohibits the development of software aids for validation, implementation, or evolution of the specifications.

If system builders are to derive the full benefit of the new paradigm, support must be provided for the development and maintainance of formal specifications. This support entails both suitable languages (notations) for system specification, a methodology for using specifications, and suitable software support environments to assist people in their use. A minimal, but nonetheless very useful, environment can be patterned after existing programming environments for high-level programming languages [6, 7, 4, 5]. But these environments rely to a great extent on the ability to compile and execute pieces of a program (specification) on test data as a means of determining the correctness of that program. The use of higher level specifications does not, however, necessarily reduce the cost of producing a testable implementation. To be directly testable, a specification must be automatically machine executable (compilable). This is at odds with having a notation that supports evolution and validation techniques well. The FSD paradigm relies on *rapid prototyping* [2] as a means of reducing the effort needed to produce testable implementations of formal specifications.

Since producing an adequate implementation of a specification is expected to be more costly than producing the specification itself, there is a large potential benefit from technology that can aid in the validation of a specification *prior* to the investment in a testable implementation. When the specifications are *formal*, the opportunity is present to apply Artificial Intelligence and software engineering techniques to the problem of validation. In particular, *symbolic evaluation* [3] can be performed on the specification itself to provide feedback to the specifier on implications of a given specification.

3.2 GOALS AND APPROACH

The FSD project's primary goal is to demonstrate the validity of claimed benefits from the FSD paradigm in three areas: software production, software maintenance, and end-user extensibility. Software is developed in the FSD paradigm by writing an operational specification of the desired software behavior and transforming it into an executable LISP program. Transformations are selected manually from a library or may be suggested by a knowledge-based AI assistant. Alternative transformational developments may be pursued in parallel. After sufficient transformation, the developer will judge the specification to be suitably elaborated to allow a conventional compiler to complete the task of producing object code.

As the specification is transformed, a record is maintained of the "program" of transformations that leads from the original specification to its implementation. Maintenance is performed by altering the specification, not the final implementation. In most cases, it is far simpler to express the desired alterations at the level provided by the specification. The transformation record of how the prior version of the specification was implemented will then be used to greatly automate the activity of generating an implementation of the altered specification [8].

End user software extensions are not formally different from maintenance changes. The impediment to such changes in the current paradigm is that end users understand a system at a level far abstracted from its implementation. Formal specifications are, or at least can be, written in terms of concepts familiar to the end users. When end users express desired changes in these terms, it will be possible (though by no means assured) that the changes can be implemented *automatically* by use of the recorded development history, which bridges the gap between specification and implementation.

The FSD project is engaged in the construction of a testbed for exploration of this new paradigm. The testbed constitutes an operational support environment for software design, implementation, and maintenance, that will support, over the project's lifetime, a succession of specification languages and implementation technologies. Progress is measured by increased support for the software designer in *formulating*, *validating*, and *implementing* software systems. The testbed focuses on supporting a level of support adequate for constructing useful *prototypes* of realistic software systems.

Each stage of the testbed system is being built using the prototyping environment supported by the prior stage. In this way, the project achieves leverage from its own research. In addition, several administrative applications are being maintained using the latest testbed support as an ongoing means of obtaining early feedback on the utility and quality of new support technology and tools.

In addition to demonstrating and exercising the FSD paradigm through internal use on the testbed and administrative applications, it is a goal of this project to construct the testbed in such a way that it can serve to house complementary research conducted externally, primarily by other DARPA contractors, on software specification on prototyping. The testbed should simplify the task of providing implementations of this research. Embedding these implementations in the testbed should, in turn, enhance the overall prototyping environment it provides.

3.3 SCIENTIFIC PROGRESS

Building on a technology and software base developed in several earlier DARPA-sponsored projects, a large and relatively robust testbed has been established for specification-based software development. The testbed consists of a kernel AI operating system, support software for specification management and implementation, and a generic user interface.

3.3.1 AI Operating System

The kernel of the FSD testbed is an operating system architecture that differs from conventional operating systems in three important facets:

- A *virtual objectbase* replaces the file system as the standard repository of data and means of interprocess communication. This provides a finer-grained and more sharable representation of information than does text-based encoding. The objectbase is a collection of *relations*, each consisting of a collection of *tuples* of values. Both the set of relations and the collection of tuples belonging to a relation vary over time. Applications can define new relations and dynamically add and delete tuples from relations.
- All changes to the objectbase are monitored by a *consistency manager*. Applications, as well as the operating system itself, may specify first-order logic invariants on the objectbase. The consistency manager ensures that no change is made to the objectbase if it would violate any declared invariant. A *repair* program may be attached to any declared invariant. When a repair program is provided, the consistency manager will execute it prior to rejecting an objectbase change that would violate the invariant. The repair program may suggest additions to the objectbase change that reestablish the invariant. In this way the objectbase changes from one consistent state to another. Changes to the objectbase are propagated through the repair programs until the objectbase is again consistent. An invariant, together with its repair program, is called a *consistency rule*. Consistency rules have proven useful for handling a variety of generic application tasks, including aspects of type-checking, error detection, software cache maintenance, and dynamic storage management.
- Each transition of the objectbase from one consistent state to another is monitored by an *automation manager*. Applications may specify trigger conditions on these transitions, using an augmentation to first-order logic notation. The augmented notation makes it possible to have portions of the trigger condition be relative to the "prior" state, and other portions relative to the "new" state. Whenever some binding of objects to variables in the trigger causes that trigger to be satisfied by a transition, a response program associated with that trigger is instantiated and queued. Programs in the queue are then executed, possibly making further transitions in the objectbase and adding new programs to the queue. When the queue empties, the main process is allowed to continue. A trigger condition, together with its response program, is called an *automation rule*. Automation rules have proven useful for transferring responsibility for recurring user activities to the system.

Figure 3-1 depicts this architecture. The operating system and its data reside in the virtual address space of a single workstation. Sharing of data among multiple workstations is currently accomplished by exporting portions of one workstation's objectbase and then importing that information into the objectbase of other workstations. This export/import mechanism preserves the identity of objects by locating exported objects in the importing world if they already exist there.

3.3.2 Specification-Based Software Development Support

The FSD testbed supports the construction and maintenance of Common LISP application programs. This support falls into two categories: language extensions and programming environment.

Specifications (programs) written in the testbed environment may not only use full Common LISP in their source code, but also extensions that provide access to the virtual objectbase, consistency rules, and automation rules. These programs may include compiler *annotations* of two types. The first type directs the compiler to use particular data structures, from a library of many alternatives, as the means of implementing the logical relations used in the specification. The second type provides the compiler with estimates of the runtime size of these relations. Using these estimates, together with knowledge of the cost of various kinds of access for each representation in the library, permits the compiler to optimize the code it generates as an implementation of the specification. This extended language is named AP5.

The FSD testbed presents the programmer with a view of specifications as collections of *definition* objects and *system construction* code. The definitions identify terms (function names, variable names, etc.) that are used as part of other definitions, in the system construction code, and possibly in other specifications. The construction code provides procedural instructions for whatever is necessary, beyond loading implementations of the definitions, to establish an initial environment that is an implementation of the specification.

These objects are hierarchically combined into modules through a *component* relation. The individual objects, as well as the modules, can have a variety of assertions made about them that are used by different aspects of the programming environment. Some of these assertions are *inherited* through the component hierarchy. For example, to parse the text associated with a definition or construction code object, Common LISP's

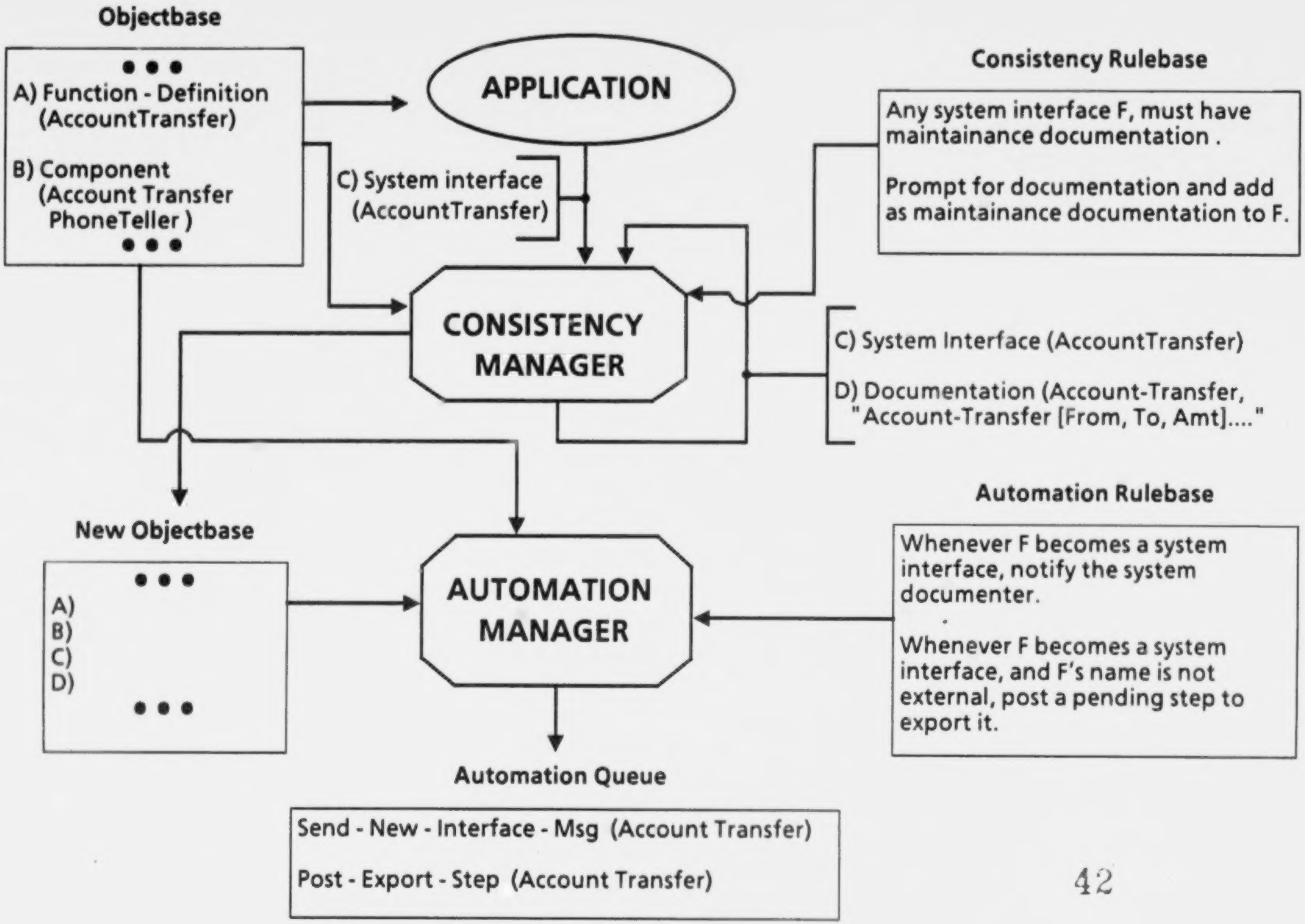
Objectbase
• • •
A) Function - Definition
(AccountTransfer)
B) Component
(Account Transfer
PhoneTeller)
• • •
APPLICATION
C) System interface
(AccountTransfer)
Consistency Rulebase
Any system interface F, must have maintainance documentation .
Prompt for documentation and add as maintainance documentation to F.
CONSISTENCY MANAGER
C) System Interface (AccountTransfer)
D) Documentation (Account-Transfer, "Account-Transfer [From, To, Amt]...."
Automation Rulebase
Whenever F becomes a system interface, notify the system documenter.
Whenever F becomes a system interface, and F's name is not external, post a pending step to export it.
New Objectbase
• • •
A)
B)
C)
D)
• • •
AUTOMATION MANAGER
Automation Queue
Send - New - Interface - Msg (Account Transfer)
Post - Export - Step (Account Transfer)

Figure 3-1: FSD Logic Engine

parser (the function READ) requires a "readtable," a "read base," and a "package."[1] This context is contolled on a file-by-file basis in conventional LISP programming environments. In FSD, it is controlled by assertions about the individual objects or modules that comprise the specification. Other assertions about modules control the *order* in which components of the module are displayed when it is edited, and the order in which the implementations of the components are installed when the software is loaded.

The root of such a hierarchy is called a *system*. Associated with the system is a *development record*, which records both the chronological development of the specification and plans for future development. This record consists of a hierarchical composition of steps, intended to reflect a goal-subgoal hierarchy. Each step includes an informal (textual) explanation its purpose, provided by the maintainer. At the leaves of this hierarchy are steps composed of collections of changes to the components of the system, such as additions of new definitions. The maintainer of a system extends the hierarchy with new steps when he chooses, and selects which step is to be "active." As he then makes changes to the components of his system to achieve the goal of the active step, the changes are automatically recorded. If he makes a change when no step is active, a new step is created and he is prompted for an explanation. The development record serves three purposes:

1. It permits the maintainer to view, or even revert to, earlier stages of his specification or earlier versions of selected components.
2. It encourages the maintainer to lay out structured plans for future development. He can then, at a later date, indicate his readiness to "handle" one of the planned steps and have it become the active step in his development, already incorporating the explanation he provided earlier.
3. It provides the necessary data for the automatic "patch" distribution facility. The maintainer of a system can choose to provide incremental upgrades to users of that system. It is normal for users of LISP workstations to save "core images" of environments into which several extensive software systems have been loaded and perhaps even used to produce data that is also resident in the saved environment. To reload one of these systems is often not an acceptable means of upgrading the environment, since it may be very time-consuming and might destroy data by reinitializing data structures. By choosing to *distribute* a step of the development record, the maintainer automatically makes available a loadable increment to the system

---

[1] The "readtable" is roughly analogous to the lexical scanner of traditional programming language parsers. The "read base" determines the radix for mapping digit sequences to integers. The "package" determines the mapping from symbolic names to LISP symbols.

implementation that users may incorporate by choosing to *update* the system in their environment.

FSD's testbed provides a *static analysis* algorithm that can be applied either selectively or automatically to the definitions that comprise the specification. This algorithm produces, as tuples in the objectbase, cross-reference information about uses of definitions by other definitions. This tool can analyze both Common LISP code and the FSD extensions. It is largely template driven, and is thus relatively easy to extend to handle new sorts of definitions and new syntactic extensions that utilize them.

The testbed also provides an automatic source code "fonter," which adds fonts to textual definitions which sharpen semantic distinctions and make the text more readable.

3.3.3 Generic User Interface

Applications written using the FSD extensions to Common LISP are automatically provided significant runtime interface support driven by the specification's domain model. Application-specific menus may also be added through declarations that associate the menu label and selection response code with specific object classes. The generic interface facilities include the following:

- *Mouse-sensitive output.* Representations of objects visible to the user (whether in "viewers" (see below) or printed by a program in a LISP interaction window) may be selected with the mouse. Any visible object may be used as input to a command being entered, used as a response to a prompt, or chosen to pop up a menu of operations that can be performed on the object.
- *Dynamic object viewers.* A particular "view" of an object (subset of its attributes and their values) may be displayed in a scrollable window. The view is kept current as the set of attributes and values in the objectbase changes.
- *Dynamic descriptions.* A collection of classifications and attribute-value pairs may be established as a *dynamic description.* The value part of each pair may be a particular object, a string pattern, or another dynamic description. Objects which satisfy the description may be displayed in an object viewer, which is kept current as the description changes. This makes it possible to "home in" on a desired object by incremental refinements to the description.
- *Pop-up viewers.* When a visible object is selected with the mouse, a pop-up window displays every generable tuple involving that object. Any object in these tuples may then be selected, producing a pop-up viewer for it. This facility makes it easy for a user to dynamically browse the objectbase in

cases where the desired "view" is not available or not known to the user. In addition to chaining from one pop-up view to another, it is possible to add tuples, delete tuples, or replace attribute values from the pop-up view.

- *Name-based searches for objects.* Objects are found whose print name matches a provided string pattern. Depending on the parameters given to the search, the space over which the search is conducted may be all instances of a class, all objects satisfying a description, or all objects in the transitive closure of a relation over some given objects (e.g., all components of some system). The string match may be applied to only the name of objects in this search space, or to the names of objects that are the values of selected attributes of objects in the search space.

3.3.4 Administrative Applications

Four administrative applications have been built and are maintained within the FSD testbed.

1. *Electronic mail* permits testbed users to receive and transmit ARPANET mail. Both incoming and outgoing messages reside in the virtual objectbase. Several users of this application have made personal extensions, via automation rules, to tailor its interface to their own preferences and to automate some of their personal message handling. A VAX-780 acts as a server for this application, to manage low-level communication with the ARPANET and to provide secure permanent storage for messages.
2. The *Alarm Clock* application permits users of the testbed to schedule processes for execution based on the real-time clock. Automation rules cannot be used for this purpose, because time is not currently represented in the virtual objectbase in a way that permits its use as the basis for triggering activities.
3. The *ISI Personnel Database* is an open-ended application to maintain personnel and contract data related to ISI. This application is currently used to maintain in the objectbase the associations between ISI employees, the "mailboxes" where they receive electronic mail, and their office assignments, phone extenstions, and project assignments. Individual testbed users make use of this application to store similar information about non-ISI employees with whom they frequently communicate. The electronic mail application uses this data, when present, to give messages the appearance of being from and to individual people rather than network mailboxes.
4. The *Scribe Interface* application provides linkage between the FSD objectbase and a Scribe document preparation system running remotely on a VAX-780. This application permits writers to build documents hierarchically from smaller pieces of text and edit at any level of the hierarchy. A document can be submitted to the Scribe compiler, and the compiled output printed on the desired output device, from the FSD testbed. The application also automatically retrieves and parses error messages produced by Scribe, rendering them as objects in the objectbase and providing an easy mechanism for positioning the editor at the site of an error in the document source.

The coexistence of these administrative applications in the testbed permits them to make use of one another's capabilities, as exemplified by the electronic mail application's use of the personnel database. In addition, the specification suppport environment uses the applications to enhance its utility. Examples of this are the provision of fonted, indexed hardcopies of specification sources (via the Scribe interface) and automatically generated electronic messages about bugs from application users to maintainers.

3.4 IMPACT

A successful conclusion of our research effort will result in more tightly integrated and evolvable systems, produced with fewer man-years of effort. The integration will arise by composing capabilities from a collection of generic, universally accessible concepts. The ability to evolve capabilities over time will result from the specificational basis of all capabilities in the environment. Both properties will contribute to productivity enhancement, both in what is traditionally thought of as system *construction* and what is traditionally thought of as system *use*.

As a side effect of this primary impact, it should become possible to produce systems with functionality beyond what is achievable from current programming environments. Evolvability will make it economical to provide community-wide or user-specific upgrades to functionality not anticipated by a service designer. Ease of sharing and specification-based software will reduce the cost of providing a given piece of functionality, encouraging service designers to provide additional functionality in their original designs.

A new DARPA-sponsored project at ISI, Common LISP Framework, will be making the initial installations of the FSD prototyping testbed during FY86 at several external sites. This will serve both as a technology transfer effort, affording the benefits of the testbed to a larger community. It is expected that several of the external sites will actually extend the testbed functionality by integrating into it additional tools that they have been developing in isolation.

3.5 FUTURE WORK

From FSD's inception, we have maintained that a technology to support specification-based design, implementation, and maintenance of software systems could not first be fully designed, and then implemented, with any chance of success. Rather it must be evolved through feedback gained from actual use of a succession of operational environments that can span the gap between current technology and the goal. Because of this, and because we believe this is the way almost any large software system should be developed, we have focused on supporting *rapid prototyping* in this environment.

The weakest aspect of the existing testbed is the specification language itself, which is an extension of Common LISP. The anticipated benefits from the new paradigm are predicated largely on gains derived from maintaining much higher-level specifications. Having categorized specification language features and the benefits afforded by each, having designed a succession of languages that incorporate these features, and having laid out the framework needed for implementation support for each language level, we are prepared to make a series of quantum steps in the specification technology supported in the tested.

Progress in the operational testbed will be characterized by distinctive plateaus, each enabling behavioral specifications to be written with less regard for the ultimate implementation hardware/software base, and each with suitable support for deriving implementations from those specifications for (at least) a Common LISP software base and the supported LISP workstation. This support will extend to maintenance of specifications by multiple maintainers and distribution of software upgrades to multiple users, at least where the maintainers and users are connected by a local area network.

The precise character of each plateau must be in some significant ways opportunistically determined by feedback from users of its predecessor. We expect to achieve plateaus characterized below during the proposed contract. We recognize that some features we use to characterize a plateau are independent of other features of that plateau, and logically could be present in an earlier plateau. The decision of when to target the introduction of a new capability is based on the need for coresident capabilities and on our estimation of the technical difficulty of providing that capability from the current state of the art.

- At the *local annotation* plateau, specifiers will be able to use a pure extension of Common LISP as their specification language. This extension will permit them to omit, selectively or totally, any specification of target language data structures to be used, and to specify logical tests and data retrievals over their data with full first-order logic, without regard to their procedural realization. The specification language will also provide for the expression of logical consistency conditions that must be maintained by the implementation, and for *data-driven* invocation of procedures (*automation rules*). Annotations added to these specifications will guide an *automatic* compiler in producing pure Common LISP code that implements them. Human implementors will select annotations with some knowledge-based assistance. This assistance will include, but not necessarily be limited to, monitoring annotations that must still be added to make the specification compilable, providing menus of annotations that are applicable to selected parts of the specification, and prohibiting selections of incompatible annotations.

- At the *independent specification language* plateau, the specifier will be using a notation that is no longer an extension of the target programming language, and that does not rely on the use of target language procedures for parts of the specification. This language will have a grammar-defined syntax. Functional and procedural abstractions will at least permit, and possibly require, typed inputs and outputs, to support greater analytical support at the specification level. The language will have an extensible typing mechanism and multiple inheritance. Implementation will still be by annotations to the specification, but many annotations will have the form of transformations, making it possible to add new annotations without altering the compiler. The power of the specification notation will still be compromised sufficiently to permit treatment of the annotations as an unstructured collection of pieces of advice.

- At the *implementation design* plateau, the specification notation will be sufficiently abstracted from the implementation hardware architecture that annotations treated as advice in terms of the specification itself will not yield acceptible implementations, probably even for prototyping purposes. The implementor, with machine assistance, will have to provide the compiler with a derivation plan for arriving at an effective implementation. Such a plan will be much like a conventional program, whose data is the specification to be implemented and whose operations are transformations on that specification. The implementor will be supported in his search for a suitable derivation plan by automated management of alternative partial plans. He will be able to selectively extend these plans and, in some cases, merge plans. The ability to reuse most of a derivation plan when an altered specification must be reimplemented is crucial to the viability of this plateau.

- At the *evolvable specification* plateau, support for performing maintenance on the specification itself will go beyond what can be provided solely by knowledge of the syntax and semantics of the specification language. This additional support may entail the specifier providing additional information about the specification, such a *purposes*, *preferences*, and *scenarios*. Our goal is to allow the specifier to describe desired changes in terms that are on a higher level than localized syntactic changes to the specification. Examples of higher level editing terms include:

- changes described as *exceptions* to the current specification;
- changes described in terms of the *set* of behaviors denoted by the current specification; and
- changes that cross the syntactic boundaries of the language, such as further "parameterizing" a current abstraction, or "generalizing" a current abstraction.

As each plateau is achieved, necessary changes will be made to the existing specifications of the FSD administrative applications, and the FSD testbed software itself, so that the continued development of these systems will take place within the environment provided by the new plateau.

REFERENCES

1. Balzer, R., "A 15-year perspective on automatic programming," *IEEE Transactions on Software Engineering* SE-11, (11), November 1985, 1257-1268.

2. Balzer, R., N. Goldman, and D. Wile, "Operational specification as the basis for rapid prototyping," in *Proceedings of the Second Software Engineering Symposium: Workshop on Rapid Prototyping*, ACM SIGSOFT, April 1982.

3. Cohen, D., "Symbolic execution of the Gist specification language," in *Proceedings of the Eighth International Joint Conference on Artificial Intelligence*, IJCAI, 1983.

4. Kernighan, B. W., and J. R.Mashey, "The UNIX programming environment," *Computer*, April 1981, 12-24.

5. Osterweil, L. J., "Toolpack-an experimental software development environment research project," in *Proceedings of the Sixth International Conference on Software Engineering*, pp. 166-175, IEEE Computer Society, Tokyo, September 1982.

6. Swinehart, D. C., P. T. Zellweger, and R. B. Hagmann, "The structure of Cedar," in *SIGPLAN 85 Language Issues in Programming Environments*, ACM SIGPLAN, July 1985.

7. Teitelman, W., *Interlisp Reference Manual*, XEROX Palo Alto Research Center, 1983.

8. Wile, D., "Program developments: Formal explanations of implementations," *Communications of the ACM* 26:11, 1983, 902-911. (Also published as USC/Information Sciences Institute, RR-82-99, 1982).

4. MAPPINGS

Research Staff:
Robert M. Balzer
David S. Wile
Martin S. Feather
W. Lewis Johnson
David J. Mostow

Support Staff:
Audree Beal

4.1 PROBLEM BEING SOLVED

Software specification, development, and maintenance continue to present an enormous problem to everyone involved with computers. We believe that the computer itself must play a far more significant role in the software development process than it does presently. The software designer's role should be streamlined to require only decision making and guidance, while the computer's role is expanded to include manipulation, analysis, and documentation. The key to such support is to capture in the machine all crucial information about the processes of specification, design, implementation, and maintenance.

We envision a future user developing a high-level specification of what he wants a program to do, and transforming the specification into an efficient program, using a catalog of (proven) correctness-preserving transformations. Most debugging and all maintenance will be performed on the specification, which will have an operational interpretation suitable for testing. Reimplementation will be necessary to tune implementations exhibiting unacceptable performance characteristics.

It is important to formally represent all crucial information used in the processes of specification, design, implementation, and maintenance. During the implementation phase of the program development cycle, the high-level constructs used to specify a program must be replaced by efficient realizations of these constructs. The Mappings project concentrated on discovering and implementing the general transformations, or *mappings*, required.

We have found that specifications are often difficult to read, despite the high-level nature of the constructs used. The essence of the problem is that the modes of communication normally used between people are considerably richer than those between people and machines. The Mappings project developed several ideas on how this communication gap can be narrowed using incremental specification.

This is the *Final Report* of the Mappings project. Indeed, the research thrusts begun by the Mappings project continue: every research thread begun in the Mappings project will receive attention under different project sponsorships, either at ISI or elsewhere. Hence, its natural termination point has been reached. In this report, we attempt to summarize the results of the entire duration of the Mappings project, as well as the recent results of the last fiscal year. In Section 4.4 we detail the legacy of the research results to current research efforts.

4.2 GOALS AND APPROACH

The primary goal of the Mappings project was to represent programming knowledge sources in terms of mappings between specifications and programs. These mappings fall into two broad categories: mappings for understanding the specification better and mappings for implementation of the specification.

We developed a specification language, called Gist, which formalizes some of the expressive constructs commonly used in natural language specifications. These constructs include nondeterminism, descriptive reference, historical reference, constraints, and demons. A Gist specification describes the behavior of both the program and its environment; this provides a natural way to specify embedded programs that interact in complicated ways with their environment. Gist's expressive power makes it possible to specify *what* behavior a program should exhibit without saying *how* it should be implemented.

4.2.1 Mappings for Optimization

Since its inception the Mappings project addressed the problem of representing programming knowledge in terms of our specification language, Gist, by discovering correctness-preserving transformations for implementing Gist's high-level constructs in terms of the lower level constructs used in more conventional programming languages. For each construct, our research accumulated: *implementation options* for converting an instance of the specification construct into a more efficient expression of the same behavior in terms of lower level constructs; *selection criteria* for deciding among these options; and, *mappings* to achieve the implementation options via sequences of correctness-preserving transformations. Recently, more emphasis was given to mappings for optimization of these implementations.

4.2.2 Mappings for Understanding

One's understanding of a specification should *evolve* rather than arise all at once. The power of prose descriptions derives, in part, from reliance on a reader's *incremental* understanding as he goes through a document sentence by sentence, subsection by subsection. We believe we can convey such incremental understanding of specifications in terms of ***high-level editing*** commands that formally describe the evolution of Gist specifications; viz. ***refinement*** of modeling detail, ***generalization*** of functionality, ***introduction of exceptions*** into idealized specifications, and ***restriction*** of scope or functionality. Our Mappings project captured some of these methods as ***formal*** methods, thus permitting the specification development to appear much more natural to future readers of the specification. Unlike the ***mappings for optimization*** these ***mappings for understanding*** often ***change*** one specification into one with observably different behavior.

A second means for enhancing understandability is to allow much more *expressive* modes of specification than are presently used. Generally, the formal languages used to describe program behavior to machines (i.e., programming and specification languages) are extremely restrictive and primitive. For example, most journal articles contain tables, grammars, equations, graphs, and other specialized notations to communicate how a program works or what it does. There is no apparent reason why computers cannot begin to understand these richer, higher level modes of expression. We have coined the phrase *locally formal notations* to describe the specification technique we espouse, to emphasize that although the notation is self-consistent, it can only be interpreted in context. That context establishes a meaning and allows the notation to be concise. Of course, these notations must be mapped onto conventional Gist semantics internally.

4.3 SCIENTIFIC PROGRESS

Four major themes dominated the Mappings project's development. The original charter was to develop correctness-preserving *transformations from Gist into LISP*. Later, the project attempted to arrive at *VLSI designs* transformationally from a Gist specification of the intended functionality. Throughout, the formalization of the *semantics of Gist* and the semantics of implementation of a Gist specification itself were necessary, if only to convince ourselves of the validity of the mappings developed. Most recently, issues surrounding *mappings for understanding* the specification dominated the research interests of the group.

4.3.1 Transformations from Gist into LISP

Considerable progress was made in the development of high-level transformations:

- **Transformations to remove Gist high-level constructs.** We developed some high-level transformations which allow us to remove nondeterminism, to eliminate special cases of recursively derived relations, and to convert references to historical information into code which first saves the "literal" information while it is available (i.e., current) and later retrieves it when needed [11].
- **Transformations for rapid prototyping.** As an aid to understanding the specification, i.e., to *validate* that its behavior is what was intended, expedient implementations can be constructed by choosing adequate implementations for Gist high-level constructs. These choices need not be optimal, but merely fast enough to examine behavior on a small number of test cases. We have called the use of transformations for such a purpose, *transformations for rapid prototyping* [5], although they differ only in that their selection criteria need not be developed fully.
- **Transformations to remove perfect knowledge assumption.** Gist's perfect knowledge assumption allows universal and existential quantification over a virtual global database of arbitrarily complex relations. Implementation of such relations requires a theorem-proving mechanism for testing suitability of proposed realizations. We experimented with a small set of implementing transformations and constructed a preliminary theorem proving mechanism to test their feasibility [12].
- **Paradigmatic transformations**: transformations which capture common high-level optimization techniques. Our most successful example of such transformations was "memoize," a transformation which introduces a data structure for caching information and causes subsequent retrieval of cached values for previously computed arguments, rather than recomputation. The development of this transformation was particularly interesting in that the concerns which went into producing the transformation seem to typify the thought processes of a human implementer when optimizing functions. First, one must decide when it is locally worth saving information, rather than recomputing it; then, whether the scope of the cache makes the savings worthwhile. Finally one must arrive at an overall caching strategy to deal with the interactions between multiply-cached functions. The memoize transformation uses a recursive descent, analytical technique to make decisions involving locality and multiple interactions, guided by heuristics and user-interaction to solve the scoping problem [15].
- **Will decomposition into local formalisms.** Several years ago, we implemented a large subset of a language called Will, an automatically

compilable subset of Gist. (This subset was formed by omitting Gist's highest level constructs.) We use this language as the target language for our Gist mappings and intend to use it as our high-level programming language for system development. Recently, this compiler was entirely rewritten as a set of local formalisms defined transformationally in terms of one another. In particular, transformations from Will into Will were used to define concepts which were simply syntactic extensions of a more substantial kernel. Then Will to LISP transformations are invoked, transforming to an inefficient version of LISP. Finally, context sensitive LISP-to-VLISP transformations are used to map into a more efficient implementation. *This use of local formalisms to decompose the Will compiler is a major joint scientific achievement of the Mappings project and the NSF sponsored project, Transformation-Based Maintenance, in the current fiscal year. It is reported in [17].*

4.3.2 VLSI Designs

The Mappings project attempted to apply the same transformational technology to VLSI design as to software design. It is clear that many of the complexity concerns are shared between the disciplines. In some sense, the software dilemma has been bequeathed to hardware designers through VLSI: they can now conceive of building into hardware, systems that are so complex they could only have been implemented in software before VLSI! The Mappings project had considerable success in capturing real hardware designs in terms of the methodology. These designs were confined to paraphrasing designs previously worked out by VLSI experts. Our accomplishments include the following:

- **VLSI as programming constructs.** VLSI primitives were modeled as constructs in conventional programming languages; hence, they were made amenable to program transformation. A germinal transformation system was implemented to illustrate the applicability of the model to a real example [13, 14].
- **Systolic array transformations.** A set of symbolic transformations was designed to translate specifications (written in a canonical style) into systolic array processing primitives (a widely used subtechnology of VLSI) [10].

4.3.3 Semantics of Gist

Over the past several years we gained considerable experience with the design of medium-sized specifications in Gist. The Mappings project played a major role in the design of Gist: in order to understand the validity of transformations, we formalized Gist's semantics. This led to greater understanding of the specification process and its relationship with both the domain being modelled and the intended implementation environment.

Gist specifications are unusual in that the behavior of an entire system and its environment is specified: every independent agent who could affect the behavior of the system must be modeled to the extent that he can affect it. This means that every person and every piece of hardware in the imagined system must be modeled. It is then the task of the implementor to implement a portion of the specification as a computer program.

- **Implementation Specifications.** We developed a formal definition of the meaning of "implementation" of a Gist specification, in which a "specification of the implementation" must be provided separate from the Gist specification of the functionality of the system. Such a specification partitions the world into the "system to be implemented" and the "environment in which it will be implemented." This partition also describes the interface between the two portions; i.e., the limits on one portion regarding control or information availability in the other portion. Any implementation which obeys the interface limitations is said to be *compliant*. A *correct implementation* imposes different requirements on these two domains. The environment is generally autonomous: the system must react gracefully to *all* nondeterministic activity from the environment. On the other hand, the nondeterminism of the system portion of the specification can be exploited for efficient implementation: *any* behavior from the nondeterministically specified set is acceptable [6].
- **Responsibility.** We dealt with the problematical issue of establishing the appropriate set of interface limitations on an implementation. The basic issue is transforming a global database and globally visible process activity into local databases of processes with appropriate communication protocols. The key to the solution of this problem has been the development of the notion of *responsibility* of agents for the maintenance of constraints in the implementation. The idea is basically simple: if an agent is responsible for the maintenance of a constraint, it must be impossible for other agents to manage to produce a situation in which the constraint is violated. Some behaviors cannot acceptably be partitioned into the responsibility of single agents. In these cases, implementation consists of substituting protocols and subsequent responsibilities of the interacting agents [7].
- **Specification Granularity.** Another important discovery we made during course of the Mappings project concerns the domain being modeled in the specification [8]. An especially confusing aspect of specifications concerns the *granularity* of the model. We have discovered that this confusion arises because there are three different axes of granularity of modeling normally involved. *Structural granularity* deals with the amount of detail the specification reveals about each individual state of a process: what basic objects and relationships between them are to be modeled. Modeling always involves abstraction away from some details. *Temporal granularity* concerns the amount of detail modeled about activities in the original domain. *Coverage* deals with the range of possible behaviors permitted by a specification. For example, in the real world some set of simultaneous events may be impossible, yet the modeled world may permit the behavior. This involves restricting combinatoric effects of the other two types of

granularity. Typically, specification refinements require dropping to a more detailed level of granularity. If a user is unable to record the evolution of the specification in terms of these layers of granularity, specifications will be difficult to read and understand by future developers and implementors of the specification.

4.3.4 Mappings for Understanding

Understanding the *evolution* of a specification is often the best way to understand what has been specified. Because this crucial information is ommitted from all present-day specification languages, specifications are hard to read. In particular, we have recognized for some time that numerous implementation decisions enter into the actual functionality specified for programs [1]. For example, any portion of a specification that is present to report error conditions to the user, or take actions based on error conditions, could not possibly be a part of the specification of the *ideal* behavior of the system--why would an ideal system make errors? Hence, at some point in the evolution of the system, someone realized that the particular implementation that would be used would necessitate dealing with such imperfect behavior. Of course, systems are never ideal: finite resources of both space and time force less than perfect implementations (e.g., finite stacks, heuristic game playing programs, and out-of-paper status flags). We believe that the best way to explain such a specification is as an *incremental* modification to the original ideal specification.

- **Identification of high-level editing categories.** We identified several categories of high-level editing steps to describe how specifications actually come about, such as refinement, generalization, specialization, exception introduction, implementation decision, and redundancy introduction. We experimented with these categories as *informal* indications of the development of actual programs using the Develop System, now in use in the our program development environment, the FSD testbed [4]. It records design decisions made in the course of developing a program. Develop encourages the developer to classify each design decision into one of these categories and document it in English.
- **Support for model modifications.** Several of the high level editing categories were sufficiently well understood that practical commands to change the FSD domain model were designed and implemented. Furthermore, the commands made some attempt to *correct* existing data stored under the outdated model, resorting to user interaction when ambiguity masked the appropriate decision. *This is a major scientific achievement of the Mappings project in the current fiscal year, and is reported in [3].*
- **Scenario-based specification design.** Gist specifications are analogous to other process-based languages in that they specify the actitivy of different agents in their entirety. The notion of *scenarios* is being developed to give

an orthogonal cut at specification, much as a data-flow description gives an alternative viewpoint on conventional program structure. Scenarios describe the allowable history of states of individual objects or agents in a system with little detail about possible interference with other objects or agents. The process of integrating scenarios becomes key, to deal with the interference and consistency issues. A prototype system, called Odette, was developed to experiment with scenarios and to provide design-level support for their development and manipulation. *This is a major joint scientific achievement of the Mappings project and the Explainable Expert Systems project in the current fiscal year, and is reported in [9].*

4.4 IMPACT

The Mappings project was supportive of the transformational implementation software development paradigm [2]. Initially, this paradigm will dramatically improve programmers' ability to maintain software systems; ultimately the entire lifecycle from design through specification, implementation, debugging, and maintenance will be streamlined. This will allow programmers to more rapidly produce and maintain software systems and will make feasible the construction of larger, more complex systems with enhanced functionality and flexibility. Of course, a major goal of the Mappings project was to facilitate system maintenance by providing system designers with modes of expression that make specifications more easily understood and hence more modifiable. In addition, this project has helped to codify the knowledge needed by programmers to convert high-level specification concepts into efficient implementations via mappings. This knowledge would also be useful for programmer education independent of its mechanized application via transformations.

Each of the research threads described above is continuing under different funding auspices or at different institutions. In particular:

- Although we laid the groundwork for some of the high-level mappings from Gist into Will, we found that the lack of low-level transformations and the lack of implemented mappings for rapid prototyping present an enormous bottleneck to the development of the FSD system in itself. Hence, the FSD project itself will focus on some quite utilitarian goals to provide the Will-specific mappings and mapping technology needed to produce a usable system in the near term. Longer term use in FSD of Gist mappings to remove specification freedoms is doctrine.
- Our VLSI accomplishments demonstrate the applicability of the transformation technology to the VLSI domain; in fact, others began to develop this technology several years ago [16].
- The understanding conveyed by the high-level editing commands and locally formal notations--to both human readers and automatic analysis tools--will

form a basis for *evolving and adapting* software. In fact, interest in these high-level mappings for understanding began to dominate the research goals of the Mappings project. Hence, concerns about both Gist semantics and evolutionary descriptions of specifications have been subsumed by a (larger) project funded by Rome Air Development Corporation to develop a Knowledge-Based Specification Assistant.

REFERENCES

1. Balzer, R., and W. Swartout, "On the inevitable intertwining of specification and implementation," *CACM* 25, (7), July 1982.
2. Balzer, R., C. Green, and T. Cheatham, "Software technology in the 1990's: using a new paradigm," *Computer*, November 1983.
3. Balzer, R., "Model Based Maintenance of Domain Structures," in *Proceedings of the 9th International Joint Conference on Artificial Intelligence*, IJCAI, 1985.
4. Balzer, R., "Living in The Next Generation of Operating System," in *Proceedings of the 10th World Computer Congress, Dublin*, IFIP, September 1986.
5. Feather, M., "Mappings for rapid prototyping," in *Proceedings, ACM SIGSOFT Software Engineering Symposium on Rapid Prototyping*, 1982.
6. Feather, M. S., Correctness and compliance criteria for implementation validity, 1984. In preparation.
7. Feather, M. S., Language support for the specification and development of composite systems, 1985. To appear in ACM TOPLAS.
8. Goldman, N., "Three dimensions of design development," in *AAAI83*, American Association for Artificial Intelligence, Washington, DC, August 1983.
9. Johnson, W. L., and J. Pavlin, Enhancing Maintainability of Object-Oriented Systems, 1986. Submitted to the Third International Conference on Data Engineering.
10. Lam, M., and J. Mostow, "A transformational model of VLSI systolic design," in *IFIP 6th International Symposium on Computer Hardware Description Languages and their Applications*, pp. 65-77, Carnegie-Mellon University, May 1983.
11. London, P. E., and M. S. Feather, "Implementing specification freedoms," *Science of Computer Programming*, (2), 1982, 91-131.
12. Mostow, J., "A Problem-Solver for Making Advice Operational," in *AAAI83*, American Association for Artificial Intelligence, Washington, DC, August 1983.
13. Mostow, J., "Program transformations for VLSI," in *IJCAI83*, pp. 40-43, Karlsruhe, Germany, 1983.

14. Mostow, J., "A decision-based framework for comparing hardware compilers," *Journal of Systems and Software*, (4), 1984.

15. Mostow, J., and D. Cohen, "Automating Program Speedup by Deciding What to Cache," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, IJCAI, 1985. August 18-23, 1985.

16. Subrahmanyam, P. A., Abstraction to silicon: a new design paradigm for special purpose VLSI systems, 1981. Submitted for publication, 9/81.

17. Wile, D. S., "Local Formalisms: Widening the Spectrum of Wide-spectrum Languages," in *Conference on Program Specification and Transformation*, IFIP Working Group 2.1, April 1986. (To appear.)

5. COMMAND GRAPHICS

Research Staff:	**Support Staff:**
Richard Bisbey II	Jan Brooks
Benjamin Britt	
Danny Cohen	
Dennis Hollingworth	

5.1 PROBLEM BEING SOLVED

The military, like the private sector, is in the midst of an information explosion. More and more computers and computer-controlled systems are being acquired, generating information in increasing quantity and detail. For this information to be useful in decision making, it is necessary for computers to assume a greater role in the storage, retrieval, analysis, integration, and presentation of data.

Information must be presented to the decision maker in ways that enhance and facilitate the decision process. The plotting of information on a graph, bar, or pie chart or the positioning of spatially related information on a map can aid in the rapid assimilation of information by the decision maker. Two-dimensional graphic displays often disclose perspectives that would not be readily apparent from a table or list of numbers. Moreover, two-dimensional displays provide a natural medium for integrating and fusing information. For example, a situation display application could graphically integrate surveillance, force, and meteorological data reported by dispersed forces to produce an aggregate picture of a particular situation. The resulting graphics display could then be used by the dispersed forces to coordinate their activities.

5.2 GOALS AND APPROACH

For graphics to play a significant role in military decision making, three basic components are needed. First, there must be a graphics system, a program that converts basic graphics primitives (e.g., lines, text, filled solids) into the order codes for a particular display device. The graphics system must meet military requirements for mobility and survivability, particularly in crisis mode. The system must also be adaptable to the changing communications, processing, and display resources available during and between crises, and must evolve to meet future command and control processing and display requirements.

Second, high-level, domain-independent display agents are needed. Display agents are "expert systems for producing graphics displays," i.e., they manage the display surface, the graphics representation, and the placement of information on a graphics display. For example, a display agent may know how to create and label a graph from a list of numbers, or create a Mercator Projection map, then draw and annotate a great circle line on that map given a list of geodetic coordinates. Display agents enable the user to create complex two-dimensional graphics displays easily, thereby removing the burden of "graphics expertise" from the user. Display agents use a graphics system as the underlying graphics support mechanism for displaying visual information.

Third, high-level, domain-dependent applications and decision aids in the form of expert systems are needed. These applications and decision aids interact with the user, the user's databases, and the display agent. The decision aids retrieve raw data, and then generalize, summarize, analyze, abstract, aggregate, and fuse the raw data, transforming it into information relevant to the user and the user's decision-making process. The decision aid also performs the semantic binding between the user's information and the high-level representations provided by a display agent. For example, a decision aid would determine the information relevant to the user (e.g., a threat envelope, a convoy of ships, a returning aircraft, or a recovery site), then select the high-level display agent representations to be used to denote that information (e.g., a transparent overlay or a labeled icon), providing the geographic coordinates of the objects for placement on a map or chart. The decision aid would not be concerned with the particular techniques employed by the display agent or the graphics system to support the various representations, nor with the production of those representations.

5.3 SCIENTIFIC PROGRESS

The three basic components, the graphics system, the display agents, and the applications/decision aids, form a natural layering, with the graphics at the bottommost layer, and the application/decision aid at the top. During a previous contract, ISI focused on the bottommost level, the graphics system component. The effort resulted in the development of a network-distributable, display-device-independent graphics system for command and control [1]. As part of that work, ISI also began research on the display agent and decision aids components, specifically, features required for the creation of geographic displays. The work resulted in the development of a Situation Display and Assessment application for retrieving and displaying Navy data.

Situation Display allowed an operator seated in front of one or more graphics display devices to pose natural language questions and commands to a distributed database manager and to receive natural language and graphics responses. Examples of natural language input included "Do any ships within 400 miles of Luanda have a doctor aboard?" or "Show me the destroyers whose radar is inoperative." Graphics responses could be in either of two forms, tabular or geographic. In geographic mode, ships and associated data, such as sensor envelopes and speed/course vectors, were automatically positioned on a background Mercator projection map.

During the development of the Situation Display, three factors were found to be particularly important in producing aesthetic geographic-based displays: intelligent placement of objects, preservation of display precedence relationships between objects, and judicious color selection.

The placement of information on the display surface proves to be critically important if the display is to be readable. In creating maps and charts, cartographers go to great lengths to improve readability, even to the extent of displacing objects from their actual geographic coordinates. For example, consider a map of the United States showing major transcontinental highways and railways. In many instances, the two run parallel to each other. If both the highway and the railway were represented at their exact geographic locations, the two would lie on top of each other, making it difficult to distinguish one from the other. A cartographer would displace one of the two some distance from the other so that both would be visible. In Situation Display, "symbol collision avoidance" algorithms were included to perform this same type of function.

Another important factor in display generation was precedence of overlays. It is often necessary to call attention to large areas. These areas might represent some meteorological phenomenon like cloud or fog cover; coverage by sensing equipment, such as a satellite footprint or a radar sensor envelope; or a path, such as an airroute or a sealane. While it is important to be able to distinguish visually the area in question, it is also important that the area be displayed in such a fashion as not to obliterate or obscure other data displayed in that same area. Overlay precedence allows one to denote these areas by making small changes to either the intensity or the chromaticity of the objects (or portions of objects) being displayed in that area (similar to placing a light-colored piece of cellophane over the area). Situation Display used overlay precedence in displaying satellite footprints, sensor envelopes, and sealanes. For example, satellite footprints could overlay sensor envelopes, which could in turn overlay ships, sea, or land. All were discernible.

Color also had important use in Situation Display. Color selection provided discrimination between nationalities for ship information. Highlighting (using blinking or color intensification) was used to draw special attention to individual ships, e.g., for responding to questions such as, "Of the ships being displayed, which ships have an ASW capability?" Colors and intensities were chosen so as not to unduly emphasize one type of object over another. Colors for background information were chosen to produce a visually apparent background.

Two important observations were made from the Situation Display development effort. First, the production of legible geographic-based displays requires a considerable amount of graphics intelligence and sophistication. In fact, the amount of programming necessary to incorporate the "graphics intelligence" often exceeds that of the application program or decision aid itself. Second, the representation mechanisms required for the Situation Display were essentially generic in nature. For example, the same annotation mechanism used to represent the location of a ship or submarine in Situation Display could also represent the location of a tank or gun emplacement on an Army Tactical display or an aircraft being recovered in an Air Force Bomber Recovery display. While the graphics display portion of Situation Display was highly tailored to a particular application domain (pictures were described in terms of specific Navy objects, e.g., Vessel Control Numbers, Unit Identification Codes, Ship Classes), if the binding of semantic information to graphic representation was done in the application program, the same graphic display package could be used by a variety of different applications.

Based on the above observations, a project was initiated to design and develop a high-level, domain-independent Geographic Display Agent (GDA) for use in DARPA's Strategic C3 Experiment. The GDA would assume total responsibility for the production of the geographic-based displays, including generation of background maps and generation, placement, and symbol collision resolution of user annotations. It would provide the application programmer with a high-level, object-oriented interface for constructing geographic-based displays, automating many of the functions a cartographer normally performs when creating geographically oriented displays. The GDA would reduce the graphics expertise required for developing aesthetic displays and would permit the programmer to focus on application issues, for which he is most qualified. The GDA's primitives would include maps, display and placement precedence control, and a general annotation mechanism, as well as traditional graphics primitives such as text, vectors, areas, lines, and swaths. Coordinates would be specified in latitude and longitude, with the GDA assuming responsibility for the transformation to screen coordinates for the particular map projection being displayed.

The first year's effort focused on defining the architecture of the overall system and the programmer's interface to the system. An initial implementation of the GDA was completed during the first year. This initial version allowed an application to specify a desired map area, place textually annotated icons on that background map, and label the display with a title, date, and classification. The GDA handled such details as background map creation, transformation of user-requested geographic coordinates into screen coordinates on the map, and placement of annotations on the map. Background map creation involved expanding user-requested map areas to fit the aspect ratio of the graphics display device; retrieving appropriate map information from a tiled, world map database; projecting and displaying the map in a form (either solid-filled or outline-only) appropriate to the graphics display device; and providing optional grid information. Annotation placement involved insuring that icons and alphanumeric information would not overwrite, and thus obscure one another, while managing object placement order so that those objects considered most important are placed closest to their desired position. In summary, the GDA provided the C2 application programmer with an object-oriented interface that allowed him to concentrate on deciding which objects to display rather than the mechanics of actually generating an aesthetic display.

The past year's work focused on extending the Geographic Display Agent's functionality. The most significant extension was the inclusion of additional display primitives. These included independent tags and markers, linears (lines and arcs), and areas (polygons and sectors). In addition, a user-specifiable placement and display precedence system was added to control objects on the screen. A transparent foreground overlay capability was also added, to permit the user to denote areas without destroying or obscuring other information within that area. Finally, a user icon facility was added to allow users to augment the built-in icons with their own application-specific icons.

A second major GDA extension was in the area of maps. The initial GDA generated solid-filled maps from a tiled, world map database. The past year's map work focused on improving the response time for creating and displaying background maps. An automatic map caching mechanism was included to eliminate recalculation of previously displayed maps. Additionally, an alternative map storage and display mechanism was implemented that utilized supplementary, pre-stored maps. Map images are pre-computed and stored in display-device-independent graphics files that can be rapidly accessed by the GDA as background maps, thus eliminating run-time map projection calculations. The GDA then places requested symbology at the appropriate locations on the resulting map image.

Finally, the underlying Graphics System and graphics software was ported to the SMI SUN, and a new device driver for the SUN color graphics hardware was developed.

5.4 IMPACT

The principal impact of this work will be felt in military environments where command mobility is paramount and a portable, network-based graphics capability is important for information presentation for the command decision process. Completion of the conversion process to the SMI SUN makes the Graphics System available in mobile environments. Development of the Geographic Display Agent facilitates development of decision aids and other application software that have a strong, geographically oriented information presentation requirement. The work also serves as a model for future tool development by clearly demonstrating the utility of developing flexible and adaptable tools that are not tied to specific hardware and that incorporate "intelligence" about particular areas of functionality to be supported.

5.5 FUTURE WORK

During the next year, the major effort will be to incorporate a laser video disk map database into the GDA. The current software map database and drawing algorithms will also be expanded to include state boundary information. This will provide an improved map environment for the GDA and C3 applications by providing more context than the current land mass outline maps.

The Geographic Display Agent will be extended to support an Area Reservation mechanism and exact locators. Area Reservation will allow the user to restrict the graphics that may appear in any user-identified geographic area. This mechanism can be used to improve visibility in the resulting picture by restricting where labels may appear. Exact locators will allow identification of precise geographic locations when schematic representations are not appropriate.

Finally, the project will focus on supporting the GDA and Graphics System through the final Strategic C3 Experiment demonstrations.

6. INTERNET CONCEPTS RESEARCH

Research Staff:
Jon Postel
Danny Cohen
Paul Kirton[1]
Paul Mockapetris
Greg Finn
Annette DeSchon
Joyce Reynolds

Research Assistants:
Alan Katz
Dave Smallberg

Support Staff:
Ann Westine

6.1 PROBLEM BEING SOLVED

The goal of the Internet Concepts Research project is to extend and enhance computer communications. The project's work is based on the ARPA-Internet, a system of interconnected computer communication digital packet networks. The rules of communication (that is, protocols) are the key element in successful computer communication. The ARPA-Internet has working protocols for communication between heterogeneous computers via an interconnected collection of heterogeneous packet networks. The ARPA-Internet currently has 1879 hosts, 491 networks, and 134 gateways.

This research covers work at several levels, involving applications protocols as well as host-to-host and gateway-to-gateway protocols. In the applications-level protocols, the focus is on new uses of the Internet based on new procedures for hierarchical naming, multimedia mail, computer mail interoperation, and the exterior gateway protocol. The basic protocols are largely complete, but a number of extensions are being explored to integrate and extend the packet network technology.

6.2 GOALS AND APPROACH

The long-term goals of the Internet Concepts Research project are to provide appropriate and effective designs for the primary user service applications in the internetwork communication environment. The designs are based on a set of host- and gateway-level protocols that provide the full range of service characteristics appropriate to a wide variety of applications.

---

[1] Visiting Scholar supported by Telecom Australia.

Our approach has been to pursue in parallel the abstract design and experimental implementation of protocols. The interaction of these activities provides valuable insights into problems and potential solutions.

Further, we divide our effort into several task areas: Hierarchical Naming System, Multimedia Mail, Computer Mail Interoperation, Exterior Gateway Protocol, and Studies, Surveys, and Specifications.

6.2.1 Hierarchical Naming System

In a large internet system, the management of names becomes a complex task. In the past, a central table of names was used. However, this simple approach has become unwieldy. To remove practical and technical constraints, to provide new functionality for new applications, and to provide for the continued growth of the Internet system, the Domain Naming System is being developed. This system provides structured names and allows subdivision of name management duties.

Our goal is to provide a design for a distributed system of domain servers that manage the access to a distributed database of names and associated data. Our approach is to create the design for the system and to review it with interested parties in the DARPA research community. As a result of feedback received from these reviews, the design will be modified. Also, we will implement a prototype domain server. The experience from this implementation will be used to modify the design and produce the final specification.

6.2.2 Multimedia Mail System

We have chosen computer mail as the application to use as a focus for demonstrating internetwork service. Within this application area, we are developing the Multimedia Mail System.

The Multimedia Mail System is an experiment to explore the future direction of computer mail systems. The primary goal is to provide specifications for computer-oriented data structures to communicate various types of data in messages, including text, graphics, and voice. Our primary interest in this mail system is in the communication mechanism rather than the user interface.

6.2.3 Computer Mail Interoperation

We are conducting an experiment in interoperation between otherwise incompatible communication systems by developing a prototype service for transmitting computer mail between the ARPA-Mail system and selected commercial mail systems (GTE Telemail, MCI-Mail, and ITT Dialcom systems). This requires (among other things) the development of techniques to address computer mail to destination addresses that may not be allowed (by syntax or semantic checks) in the originating mail system.

6.2.4 Exterior Gateway Protocol

The Internet system design allows for sets of gateways (called autonomous systems), where each set has a routing information exchange procedure (a routing algorithm) used within the set. We assume that the gateways of a set are manufactured by a single company and/or operated and maintained by a single operations center. To provide for communication of routing information between gateways of different sets, the Exterior Gateway Protocol (EGP) is defined. We have developed an implementation of EGP.

6.2.5 Studies, Surveys, and Specifications

The Internet system is now operational. However, many potential extensions to the Internet have been discussed and desired but have not been designed, implemented, or tested. Our goal is to study some of these issues and, when possible, provide designs for their eventual development.

The areas of addressing, routing, and multiplexing are particularly subtle and require careful attention. We have concentrated on these areas and have explored many options in technical discussions and memos. Our approach is to develop an understanding of these technical issues and to advocate the inclusion of general-purpose support mechanisms in the protocol specification.

In the operation and the evolution of the Internet system, it is sometimes useful to survey or measure the implementation status and performance of particular protocols. As the Internet system evolves, and as flaws and ambiguities are discovered, specification documents must be upgraded.

6.3 SCIENTIFIC PROGRESS

This year we made major progress on the domain naming system, which is now being deployed throughout the Internet. Our implementation of the user interface program for the multimedia mail system on the Xerox 1108 (in Interlisp) is now being tested by users. The Intermail experiment on computer mail interoperation has proved a success at an experimental level. The EGP implementation was completed and is now in widespread use.

6.3.1 Hierarchical Naming System

The domain name service takes a structured host name (e.g., "F.ISI.USC.EDU") and returns the numeric Internet address (e.g., "10.2.0.52"). In addition, the domain name service may have information about mailboxes and other objects. If one asks about a particular mailbox (e.g., "Postel@F.ISI.USC.EDU"), the result may be instructions to send the mail to a different host (e.g., "MAIL-CENTER.ISI.USC.EDU"). The domain server also supports inverse query resolution (i.e., host address to domain name).

Non-domain-style host names and nicknames will become obsolete as the community converts to the new distributed domain database, which is replacing the outdated HOSTS.TXT file (the central table of names). Because of this, more and more organizations are using their official domain-style names as their primary names.

We have implemented a TOPS-20 JSYS to perform host name lookups using the domain name servers. The TOPS-20 implementation includes a JSYS interface to the database and a resolver/cache. The domain name server's database includes the transport/service information from HOSTS.TXT as the Internet-specific well-known-sockets (WKS) resource records.

The domain name server is operational on USC-ISIB, USC-ISIC, and SRI-NIC. The server is a complete implementation of datagram domain name services, and can be used to load arbitrary domains. The ISIB name server has a root domain and an experimental ISI domain, plus all current .ARPA hosts. The domain name resolver for the XEROX 8010 Mesa development environment is operational, including well-known-service queries and pointer queries. This client implementation has been tested against both the TOPS-20 and the Berkeley UNIX servers. The domain name system is further described in a conference paper [1].

6.3.2 Multimedia Mail System

The transfer protocol for the Multimedia Mail System is implemented in a process called a Message Processing Module (MPM). The MPMs are responsible for the routing, transmission, and delivery of messages. Multimedia messages are created by a user with a User Interface Program (UIP). Messages so created are then submitted to an MPM for delivery.

The MPM is operational on the TOPS-20 systems at ISI and at several other sites on the ARPANET. This has made it possible for us to exchange multimedia messages with other organizations that implement UIPs. A UIP allows a user to create a multimedia message which may contain various types of text, image, and voice data and then to send the message to other hosts within the Internet environment.

At ISI we have two UIP implementations: the MMM program, developed in Pascal on the PERQ, and the MMH program, developed in Interlisp on the Xerox 1108 (Dandelion). Several improvements were made to the MMM program on the PERQ. The current version allows scanning messages which have unknown entity types or protocols, and adds a renumber command and a mailing list capability. At ISI, the TFTP file transfer protocol is used for the transfer of messages between the MMM program (on the PERQ) and the MPM on TOPS-20. We continue to send multimedia mail routinely to and from three PERQs at ISI.

The MMH program was made available for experimental use at the end of this year, and many new features are now available to the users of multimedia mail at ISI. These include scrolling of spatially controlled windows and a graph diagram of the message structure. The Interlisp-D Sketch bitmap editor was integrated into the MMH program. It provides interactive control over the editing of image presentation elements of the bitmap protocol. Support is provided for transfer of images to and from files in RFC-797 format. At ISI, the FTP file transfer protocol is used for the transfer of messages between the MMH program (on Xerox 1108s) and the MPM on TOPS-20. We are able to send text multimedia messages via the MPM from the Dandelion to the PERQ, and vice versa.

We have developed a system using a Packet Voice Terminal (PVT), connected to the Internet, as a voice server. This system enables either the MMM or MMH programs to send packets containing voice data to the PVT. The PVT then calls the user on his telephone and plays the voice through the receiver. A similar chain of events allows the user to enter voice data.

A service (called BITSRV) has been implemented that automatically converts bitmap format files to Rapicom facsimile format files. This allows bitmap files to be deposited (directly or via TFTP or FTP) into the FAX-server directory and output as facsimile images on the Rapicom machine, without any additional human intervention.

Multimedia mail demonstrations were given to many visitors throughout the year. A new version of a document describing the text and image media data structures for the body of multimedia messages was written. In addition, a preliminary specification for the spatial construct was defined.

6.3.3 Computer Mail Interoperation

The evolution of large electronic mail systems testifies to the increasing importance of electronic mail as a means of communication and coordination throughout the scientific research community. These systems include the DARPA Internet mail system, the GTE Telemail system, the MCI-Mail system, and the IEEE Compmail system. Until now these systems have operated autonomously, and no convenient mechanism has existed to allow users of one system to send electronic mail to users on another system. INTERMAIL is an experimental mail forwarding system which allows users to send electronic mail across such mail system boundaries. Users on each system are able to use their usual mail programs to prepare, send, and receive messages. No modifications to any of the mail programs on any of the systems are required.

INTERMAIL has a login account and a mailbox on each of the commercial mail systems it services. A user on a commercial system sends mail destined for a user in the ARPA-Mail world to the "Intermail" mailbox on his local commercial system. The INTERMAIL program periodically picks up this mail, determines the destination ARPA-Mail address, and turns the message over to the Internet mailer for delivery to the appropriate Internet host. In the other direction, a user on an Internet host sends mail destined for a user on a commercial mail system to the ARPA-Mail mailbox of INTERMAIL ("INTERMAIL@USC-ISIB.ARPA"). INTERMAIL periodically reads messages from its mailbox on USC-ISIB, determines the destination mail system, and sends the message using TELEMAILER, COMPMAILER, or MCIMAILER. The INTERMAIL program examines the forwarding information contained in each message to determine the destination system.

The Intermail experiment is in operation between ARPA-Mail and selected commercial mail systems. Recently we have improved the error handling, accounting, and access control. Modifications were made to MCIMAILER to make it compatible with version 2.3 of MCI Mail. We developed a simplified user interface to be used for mail forwarding. This interface allows the user to specify a destination mail system and the destination-system as fields, at the beginning of the text of the message which is being forwarded. It may be used to do one-hop forwarding (only).

Instructions prepared for using the simplified forwarding Intermail procedure were distributed to the experiment user community and an ISI Report was published [2].

6.3.4 Exterior Gateway Protocol

Early in this year we finished our work on the development of the EGP for Berkeley UNIX 4.2. This code has been given to Berkeley for distribution. At least 20 gateways in the Internet are using this implementation of EGP. The program is documented in ISI reports and an RFC [3].

6.3.5 Studies, Surveys, and Specifications

In this area we have participated in DARPA community efforts to develop the Internet and construct new demonstrations of computer communication technology. We have aided this effort by producing memos on new protocols, revising the specifications of old protocols, and coordinating the usage of existing protocols. We have participated in many community meetings and in scientific conferences. We have published many ISI technical reports, conference, and journal articles [4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18].

6.3.6 Meetings Attended/Papers Given, RFCs, ISI Reports

We participated in the Internet research program by our involvement in the Internet Activities Board (IAB), the International Cooperation Board (ICB), the Internet Research Group (IRG), and the Task Forces on (1) New End-to-End Services, (2) Applications, (3) Gateways, and (4) Interoperability. We also contributed to conferences and publications, including the following:

- Paul Mockapetris attended the National Computer Conference and participated in a panel discussion on "The World of Interconnected Networks."

- Jon Postel and Paul Mockapetris attended INFOCOM85. Jon gave a talk covering two papers titled "The DARPA Internet Protocol Suite" and "Internetwork Applications Using the DARPA Protocol Suite." Paul gave a talk on the paper titled "A Perspective on Name System Design."

6.4 IMPACT

The impact of packet-switched computer communication has been enormous. The success of the Defense Data Network (DDN) is one example of the importance of this work. Further enhancements of such systems depend on continuing research in computer network technology. The work reported here extends the previous capabilities. The Multimedia Mail effort explores new user-to-user information exchange capabilities. The Domain Naming System extends the power of the Internet system to identify resources by name by increasing the flexibility and dynamics of name management. The Computer Mail Interoperability effort extends communication capability for this popular mode of computer-based communication. The EGP implementations provide the basis for a multi-vendor gateway system. The expanding use of the Internet system requires a continuing effort to provide up-to-date knowledge of system design, specifications, and performance.

Hierarchical Naming System

The implementation of the Domain Naming System will significantly ease the administrative burden on the current name management group. It will also place the control of changes and updates to the database closer to the organizations affected.

Multimedia Mail

The potential for multimedia communication in a computer-assisted environment is great. The ability to communicate diagrams or maps and to then talk about them will tremendously increase the effectiveness of remote communication. The combination of text, speech, graphics, and facsimile into a common framework and data structure may have substantial impact on other applications as well.

Computer mail is the most significant use of the new communication capability provided by packet-switching networks. Our continuing work to extend the range and capabilities of computer mail will have important consequences for the DoD.

The power of a communication system is directly related to the number of potential communicants. For computer mail, this means that the power of a system is related to the number of people who have access to that system. Access to a computer mail system requires the use of compatible components: terminals, programs, and protocols. Our work on protocols and programs will increase the power of computer mail by enlarging the set of compatible components.

Computer Mail Interoperation

The experiment in computer mail interoperability has proven successful with the small test community. The capability to extend this popular service beyond the boundaries of the Internet may have significant impact on the options available to provide government contractors access to common computer mail facilities with government agencies. Many contractors have been granted access to the ARPA-Internet to enable contractor/government communication. It may now be feasible to direct some such contractors to use commercial computer mail systems and the INTERMAIL interconnection service.

Exterior Gateway Protocol

The EGP design and the experience gained in using it provide the basis for a multi-vendor gateway system in the Internet. This is very important, since it now permits gateways to be acquired on a competitive basis rather than being locked in to one vendor.

Studies, Surveys, and Specifications

The selection of the IP and TCP protocols by the DoD as the basis for a DoD internetwork protocol standard shows the impact of the work of the DARPA community on DoD communication systems. The development of the Defense Data Network (DDN) by the DCA is a major use of these protocols in an operational military system. This influence is demonstrated further by mounting requests for information about IP/TCP from companies interested in producing commercial products or bidding on government contracts.

Through our participation in the Internet Working Group meetings and in technical meetings with other contractors, we have successfully influenced the development of many protocols and protocol features. Our publication in conferences, journals, and newsletters extends the impact of this work to others who design similar systems.

6.5 FUTURE WORK

We will continue to work in these task areas: Hierarchical Naming System, Multimedia Mail, Computer Mail Interoperation, and Studies, Surveys, and Specifications.

6.5.1 Hierarchical Naming System

We will complete our development efforts in this area in the first few months of the next year. We will prepare the final documentation of the protocol specifications.

6.5.2 Multimedia Mail

We will continue to operate the MPM program and the MMH program. The interest and activities relating to multimedia will be focused in a new Multimedia Conferencing project.

6.5.3 Computer Mail Interoperation

We will continue to operate the INTERMAIL program. There is a new effort in the ISI Computer Center that will develop a fully supported service to replace our experimental service.

6.5.4 Exterior Gateway Protocol

This work is complete. No further work in this area is planned.

6.5.5 Studies, Surveys, and Specifications

We will continue to conduct surveys and specifications of Internet protocol implementation features, conduct numerous performance measurements, and report the results to the ARPA-Internet community.

We will produce up-to-date documents of the protocols used in the ARPA-Internet community as needed, and will manage the assignment of protocol parameters to network experimenters as needed.

REFERENCES

1. Mockapetris, P., and J. Postel, "A Perspective on Name System Design," in *IEEE INFOCOM85*, Washington D.C., March 1985. Also as USC/Information Sciences Institute, RS-85-152.
2. DeSchon, A., *MCI Mail/ARPA Mail Forwarding,* USC/Information Sciences Institute, RR-84-141, August 1984.
3. Kirton, P., "EGP Gateway Under Berkeley UNIX 4.2," RFC 911, USC/Information Sciences Institute," August 1984. Also as USC/Information Sciences Institute, RR-84-145.
4. Kirton, P., "Some OSI Experience of the ARPA-Internet," in *Proceedings of*

the Second International Conference on Introduction of Open Systems Interconnection Standards, Ottawa, Canada, May 15, 1984. Also as USC/Information Sciences Institute, RS-84-142, July 1984.

5. Finn, G., "Reliable Asynchronous Transfer Protocol (RATP)," RFC 916, USC/Information Sciences Institute, October 1984.
6. Reynolds, J., "Post Office Protocol," RFC 918, USC/Information Sciences Institute, October 1984.
7. Postel, J., and J. Reynolds, "Domain Requirements," RFC 920, USC/Information Sciences Institute, October 1984.
8. Postel, J., "Domain Name System Implementation Schedule - Revised," RFC 921, USC/Information Sciences Institute, October 1984.
9. Reynolds, J., and J. Postel, "Assigned Numbers," RFC 923, USC/Information Sciences Institute, October 1984.
10. Reynolds, J., and J. Postel, "Official ARPA-Internet Protocols," RFC 924, USC/Information Sciences Institute, October 1984.
11. Postel, J., "Multi-LAN Address Resolution," USC/Information Sciences Institute, RFC 925, October 1984.
12. Reynolds, J., "ISI-Hosts," USC/Information Sciences Institute, October 1984.
13. Leiner, B., R. Cole, J. Postel, and D. Mills, "The DARPA Internet Protocol Suite," in *IEEE INFOCOM85*, Washington D.C., March 1985. Also in *IEEE Communications*, March 1985. Also as USC/Information Sciences Institute, RS-85-153.
14. Postel, J., "Internetwork Applications Using the DARPA Protocol Suite," in *IEEE INFOCOM85*, Washington D.C., March 1985. Also as USC/Information Sciences Institute, RS-85-151.
15. Butler, M., J. Postel, D. Chase, J. Goldberger, and J. Reynolds, "Post Office Protocol - Version 2," RFC 937, February 1985.
16. Reynolds, J., and J. Postel, "Assigned Numbers," RFC 943, USC/Information Sciences Institute, April 1985.
17. Reynolds, J., and J. Postel, "Official ARPA-Internet Protocols," RFC 944, USC/Information Sciences Institute, April 1985.
18. Mogul, J., and J. Postel, "Internet Standard Subnetting Procedure," RFC 950, USC/Information Sciences Institute, August 1985.

7. STRATEGIC COMPUTING INFORMATION SYSTEM

Research Staff:
Danny Cohen
Leroy Richardson

Support Staff:
Jan Brooks

7.1 PROBLEM BEING SOLVED

The success of the Department of Defense's Strategic Computing research program depends on the easy dissemination of program-related information to a geographically distributed research community. The Strategic Computing Information System (SCIS) project, now completed, has provided an automatic and efficient system which can be used for information exchange between the Strategic Computing personnel and the research community at large.

7.2 GOALS AND APPROACH

The goals of the SCIS project were:

- to communicate with users via several commercial electronic mail systems;
- to supply automatic retrievals of documents in response to requests;
- to provide control over the access of "public" and "private" document files;
- to provide means for manual communication via electronic mail, as needed; and
- to run on an inexpensive personal computer.

The SCIS programs provide a means for users of commercially available mail systems to make requests, by means of electronic mail to a mailbox on each of the systems, that will be fulfilled automatically by programs running on a PC. Such mail can include requests for documents, requests for indexes to documents, and requests for special attention. The PC is not required to support on its own an electronic mail service. Rather, the PC functions more closely as an automated secretary, reading requests for various kinds of documents or index files and sending back replies on the same external mail system containing the requested documents or files. Requests for special attention are answered with an acknowledgment reply and are saved in a special set of files for manual treatment by staff personnel.

The basic cycle of operations that the PC performs under the SCIS programs is as follows:

1. Dial into a particular mail system (e.g., MCI-Mail) and log in as the mail system user providing service for users on that system (e.g., SCIS).
2. Read the mail for the user into a set of files on the PC.
3. Send any mail pending for users on that mail system, containing replies to requests.
4. Log off the particular mail system.
5. Read and parse the files from that mail system, looking for requests. For each request, construct a response message (i.e., a new file) that contains either the document or file requested or a response that it could not be found, along with the network mail address of the recipient(s).
6. Wait some time (so as not to be dialing out continuously).
7. Select the next particular mail system (e.g., TeleMail), go back to Step 1, and proceed from there.

The above steps are done for some set of mail systems; currently, the set includes MCI-Mail, TeleMail, and ARPA-Mail.

A diagram of the flow of major data among the programs is shown in Figure 7-1; it shows the way in which requests, posted on a mail system serviced by SCIS, are fulfilled.

The particular requests that users of SCIS would make are stated using a simple syntax rather than unstructured narrative text, so as to facilitate automatic processing. Text that does not conform to the syntax is generally ignored, rather than causing the request to fail (thus, parenthetic comments and the like are tolerated rather than being treated as a "user error").

Users' requests may include an access control code or identification. Access control is maintained per user. Documents requiring an access control code (i.e., "private" documents) are kept as files in special file directories, according to the access permitted. Each access control code specifies which file directories may be accessed to fulfill the user's requests for documents. Private documents to be sent to a user are sent only to electronic mail addresses that are specifically associated, by the SCIS staff, with the user's access control code.

```
|vvvvvvvvvvvvvvvv|         -------------         |-----------------|
| Incoming mail  | Tel Co  | Mailer    |         |  Received-mail  |
|      from      |vvvvvvvv>|           |-------->|      files      |
|  mail system   |  line   | program   |         |      on PC      |
|vvvvvvvvvvvvvvvv|         -------------         |-----------------|

|----------------|         -------------         |-----------------|
| Received-mail  |         | Request   |         |    Send-mail    |
|     files      |-------->|           |-------->|      files      |
|     on PC      |         | parser    |         |      on PC      |
|----------------|         -------------         |-----------------|

|----------------|         -------------         |vvvvvvvvvvvvvvvvv|
|   Send-mail    |         | Mailer    | Tel Co  |  Outgoing mail  |
|     files      |-------->|           |vvvvvvvv>|       to        |
|     on PC      |         | program   |  line   |   mail system   |
|----------------|         -------------         |vvvvvvvvvvvvvvvvv|
```

Figure 7-1: Flow of Major Data Items

7.3 SCIENTIFIC PROGRESS

The SCIS system was adapted from programs written for the MOSIS service and for cross-network mail forwarding by the Internet project.

The initial version of SCIS, written in Mainsail and SAIL, ran on TOPS-20 to provide retrieval service directly only to users making requests via the ARPANET. It did not communicate via dial-up lines, provide access control, or run on a PC. It also depended almost exclusively on the electronic mail system on the machine on which it ran for much of its mail delivery.

The new SCIS programs running on a personal computer, are written in C, a much more widely available programming language than either Mainsail or SAIL. They were derived in substantial part from the original Mainsail and SAIL code via a combination of systematic textual substitution using a programmable editor (TECO) and judicious manual editing of the results.

To provide features of the Mainsail and SAIL environments not directly provided in the C environment, two major programming aids had to be written: a text-string package and a procedure library package. The text-string package provides a means to allocate and deallocate memory for text strings in a nearly automatic manner by means of calls to the C library memory allocation routines. The procedure library package provides a set of procedures that mimic those provided by the Mainsail run-time library. These two packages together substantially eased the conversion of the Mainsail and SAIL programs to the C environment.

With the incorporation of the two major programming aids into the SCIS programs, most of the conversion of the original Mainsail and SAIL code was complete. Additional programming work was needed to:

- allow the systematic processing of requests from multiple mail systems;
- eliminate the dependence of the programs themselves on a local mail service;
- provide control of access to "private" documents.

The first of these tasks was accomplished by the automatic addition to incoming messages, by the programs receiving them, of information specifying the mail system from which they were received. The request processing program was then changed to write its replies to files with names specific to that system. Each mailing program would then send as mail only those files with the correct file names.

The elimination of the dependence on a local mail service was accomplished by patiently tracking down those dependencies and substituting direct file manipulations.

Access control was achieved by taking advantage of the UNIX login procedures and files, that provide a simple mechanism for storing and retrieving access information for a user. To access a "private" document, a SCIS user must supply in the request a user identification and password. If the user and password information match the information stored in the login database file, then SCIS will access a second file for the remaining stored information. The specific information stored for a user are the directory search paths to use for looking up information and index documents, and the specific pre-arranged electronic-mail network and network mailbox to use for sending back the documents requested. Thus control is maintained over both the access to documents via password protection, and where the accessed documents are mailed.

7.4 IMPACT

Systems such as the SCIS will allow vendors of computer services to provide many of those services automatically via electronic mail. SCIS in particular provides information retrieval; with additional work, dependent on the application, other services could readily be provided (as in the case of MOSIS, which provides VLSI fabrication services). The design of SCIS allows these services to be provided by programs running on a small machine, such as a PC. These features together should encourage the development of many such electronic-mail-based applications, thus furthering the development of electronic commerce.

8. WIDEBAND COMMUNICATION

Research Staff:	Research Assistants:	Support Staff:
Stephen Casner Alfred Beebe Jeff LaCoss David Walden	Ehoud Haberman	Jan Brooks Jerry Wills

8.1 PROBLEM BEING SOLVED

The Wideband Communication (WBC) project has its origin in efforts to establish an integrated communications network for the military. The critical importance of communications in military operations underlies the attempt to unify systems supporting the full range of communications media, including data, voice, video, facsimile and others. ISI has participated in the development of communications protocols and in the development of a demonstration packet voice system using the Wideband Packet Satellite Network. The focus of the Wideband Project is currently on the development of a packet video system.

The potential of Packet Switched Networks [6] for integrated communication has been recognized since the DARPA Network Secure Communication program demonstrated integrated voice and data communication on the ARPANET, a terrestrial packet-switched network. This practical demonstration was supported by an economic analysis of the costs and benefits of packet switching and the major competing communication technologies. This study by the Network Analysis Corporation [5] clearly identified packet switching as the preferred technology for integrated voice and data communication.

Although its data rate is limited to 56 Kbps (kilobits per second), the ARPANET did suffice as a testbed for packet voice. However, it lacked the capacity to support a large volume of voice traffic or to serve as a communication channel for higher bandwidth media such as video or simple high-volume data transfers.

Initial experiments with a satellite packet switched network utilized the Atlantic Packet Satellite Network (SATNET) which provided a 64 Kbps data rate [4]. In order to meet the need for a packet-switched network with adequate capacity for high-bandwidth communication, the DARPA/DCA Wideband Packet Satellite Program was established to create the Wideband Packet Satellite Network (WBNET). Utilizing a single geosynchronous Westar satellite to link 10 sites in the continental United States, the WBNET provides a raw data rate of 3.088 Mbps (megabits per second) for experiments with large-scale packet voice, packet video and other high-data-rate applications. While the adaptation of ARPANET and SATNET technology for voice communication on the WBNET represents a step in the maturation of packet voice, the WBNET provides for the first time a testbed for the demonstration of packet video.

While packet-switching technology underlies both the ARPANET and the WBNET, satellite communication networks have certain distinct features. Satellite networks enjoy much higher data rates and an inherent broadcast nature but they suffer from longer transmission delays and higher bit error rates. The WBNET is also powerful enough to provide a high-bandwidth link between geographically distributed local area networks (LANs) for the first time. The development of packet communication for the Wideband Network seeks to exploit these advantages and compensate for the drawbacks.

ISI's Wideband Communication project, as one of several participants in the DARPA/DCA Wideband Packet Satellite Program, is involved in the development of packet voice, packet video, and high-data-rate transmissions over the Wideband Network. The primary research focus of the Wideband Communication project is to evaluate the potential of packet video. Given the packet-switched communication channel provided by the WBNET, can existing video conferencing technology be adapted to achieve successful packet video? The secondary focus is to support efforts to demonstrate the feasibility of large-scale packet voice and the feasibility of other high-data-rate communication using the WBNET.

8.2 GOALS AND APPROACH

The basic approach of the ISI work is to build operational systems for the WBNET which can give a practical demonstration of the potential of high-bandwidth packet-switched networks to support integrated voice, video, and data communication. Mature technology is utilized for these systems where feasible, but new technology is developed when needed. Experimentation is used to optimize the system performance for the WBNET environment. In the process, useful information can be developed for tuning the Wideband Network itself to the needs of voice, video, and data communication.

ISI has developed the STNI (Switched Telephone Network Interface) which, along with Lincoln Laboratory's PVT (Packet Voice Terminal), provides a pathway for voice communication over the WBNET that can be accessed from the standard telephone system. ISI is currently building a complete digital video system for the transmission of packet video over the network. ISI participates in joint efforts to maintain and improve the reliability and performance of the WBNET, including problem identification, troubleshooting, and maintenance of the ISI node of the network.

Packet Voice. The demonstration of large-scale packet voice requires the satisfactory performance of equipment already developed and sufficient experience with high-volume voice traffic to identify and solve unsuspected problems. A potential user community has been developed as a base for this high volume of voice traffic. However, poor network performance has so far restricted voice traffic to minimal levels and has made high-volume traffic impossible. These performance problems are being attacked by a general upgrade of basic equipment for the network nodes. In addition, a new generation of STNI technology is being developed by BBN Laboratories for systematic integration of packet voice equipment into the BBN Voice Funnels. The Voice Funnels serve as gateways to the Network. The number of available STNIs will double and their operational reliability will be improved.

Packet Video. While large-scale voice certainly requires the higher bandwidth of the WBNET, packet voice techniques were earlier worked out on lower bandwidth networks (ARPANET and SATNET). Packet Video is the first new communication application on the Wideband Network, and the first one that really requires the available bandwidth.

The basic steps in ISI's packet video program are:

1. Design and construction of an operational video system, including hardware and software.
2. Installation of two copies of the video system for video communication experiments between ISI and a second site (currently Lincoln Laboratory).
3. Experimental tuning of the video system (and possibly the network) to achieve optimal performance.
4. Evaluation of the compatibility of real-time video with the Wideband Network and development of the information required to make the best fit between the two.

Although commercial video bandwidth compression systems exist, they lack the programmability necessary to tailor them to the WBNET environment. This has dictated the development of a new system combining existing technology with new special-purpose hardware. The ISI system couples a commercial digital video system, for camera input and image display, with a special ISI-designed-and-built processor for video compression. The ISI Video System is highly programmable and has the flexibility needed for a wide range of experimentation.

Design of the Video System has been driven by a number of elements [2]. Constraints are imposed by the nature of the communication channel, by the nature of digital video, by the power of existing image-processing techniques, and by the capabilities of current hardware. The design revolves around the fact that standard video (512x480-pixel pictures at 30 frames a second) produces a data stream of about 60 Mbps and the total satellite channel capacity is only 3.088 Mbps. The best techniques available for the necessary data compression are based on the Discrete Cosine Transform and frame-to-frame differencing. Although these methods are mostly well defined, considerable flexibility is required to tune the image processing to achieve desired picture quality while maintaining a low data rate. The design must also contend with the fact that channel performance is subject to sudden degradations. Fluctuating channel capacity, transmission delays, bit errors, etc., require strategies for graceful degradation of the picture in the event of information loss. So, while the system must be capable of the necessary data compression, a great deal of flexibility is necessary. For this reason the system should be as programmable as possible.

8.3 SCIENTIFIC PROGRESS

8.3.1 Wideband Network System Upgrade

While the primary focus of the ISI Wideband Project is specific communication applications, such as voice and video, the satisfactory performance of the Wideband Network itself is critical to the success of these applications. In the past, issues of system performance have been the primary concern of the Wideband Task Force, which has continuously monitored the system and attempted to squeeze the best possible performance out of the existing hardware and software.

A history of inadequate performance, including frequent and unexplained system crashes and extensive equipment downtime, has finally led to plans for a system upgrade. This will affect basic elements of the network node which consists of an Earth

Station (antenna and RF converters), ESI (Earth Station Interface) and Satellite IMP (Interface Message Processor). The upgrade will include:

- larger 7 meter antennas and more reliable RF converters,
- new BSATs to replace the PSAT network node IMPs.

Replacement of the remaining 5 meter antennas with 7 meter antennas will provide a stronger received signal. The old IMPs, called PSATs (Pluribus Satellite IMP), will be replaced by new IMPs called BSATs (Butterfly Satellite IMP) which utilize BBN's multi-processor Butterfly Computer. New versions of the Linkabit ESI will also be installed. A combination of new hardware and new software represents a broad attempt to correct known problems.

8.3.2 Packet Voice

In addition to the network upgrade, simplification and consolidation of packet voice equipment will improve voice communication performance. ISI has written the software for the BBN adaptation of the original ISI STNI. The new STNIs will connect directly to the BBN Voice Funnels through a high-speed serial interface. The original STNIs connected to the Lincoln PVTs using a parallel interface in a configuration called the Lexnet. The Lexnet could connect either to the PDP 11/44 Gateway or to the Voice Funnel Gateway both of which provide entry to the Wideband Network. In the new system a program which performs the equivalent of four Lincoln PVTs will run in the same Butterfly Computer running the Voice Funnel. This will increase the available number of STNIs and make use of the greater capacity of the Butterfly to increase the functionality of the PVT program as compared to the Lincoln PVTs.

Additional work on improving STNI performance for long distance calls has involved increases in the signal level to the touch tone decoder for more reliable long distance reception of touch tones. Output volume level has been further increased to compensate for transmission loss to the listener during a long distance connection.

8.3.3 Packet Video

This section gives an overview of the Packet Video system design and a brief progress update. Following this section are discussions of various aspects of the system.

Design Overview. The Video System design can be outlined as follows. There are three sections: a front end video I/O section (the commercial system); an image-processing section (composed of a processor from the commercial system and the new ISI processor); and a data transmission section to connect to the network (see fig. 8-1). The video I/O section consists of a camera, display monitor and image data buffer for images scanned in from the camera or for display. The image processing section includes the system controller, which manages all three sections, and the data compression processor. Pictures are input from the camera, compressed and sent to the network node for transmission. Incoming data is received from the network node, expanded and displayed. The data transmission section includes a high-speed data link which exchanges data between a compressed-data buffer, used by the image-processing section, and a message protocol program which formats the data for the network.

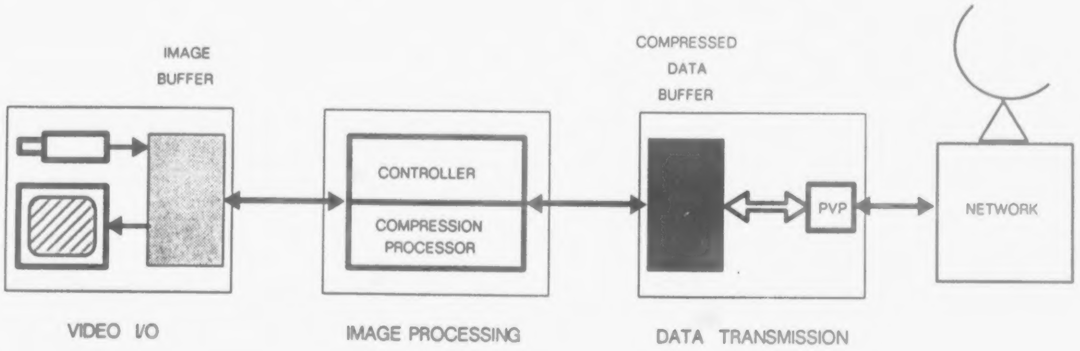

Figure 8-1: Video System Overview

The image-processing section employs a two-phase data-compression scheme and a single-phase data-expansion scheme. All the processing is done block-wise on 16x16-pixel sub-blocks of the image. Data compression performs inter-frame differencing followed by intra-frame compression. The inter-frame differencing algorithm compares consecutive frames and determines which blocks in the new frame need to be updated. A tradeoff must be made between the rate at which pictures are input from the camera and the amount of update required for each frame. Both picture quality and the fidelity of motion tracking depend on the effectiveness of this tradeoff process. Intra-frame compression is then performed on the sub-image of each frame identified for update: the image data is transformed block-wise by the Discrete Cosine Transform to produce frequency coefficients which can be compressed in a systematic way. The resulting compressed data is then transmitted. Data expansion involves only intra-frame processing which transforms compressed, frequency-domain data back to image data using the inverse Discrete Cosine Transform. The resulting image blocks are then used to update the picture being displayed.

The data transmission section is responsible for the two-way exchange of data between the compressed data buffer and the Voice Funnel which serves as a Network Gateway. The Voice Funnel program runs on several processors of a BBN multi-processor Butterfly Computer. The data link itself is a physical pathway for high-speed (1 or 2 Mbps) transmission. At one end of the data link, and running on another processor in the Butterfly, is the message protocol program, PVP (Packet Video Protocol). At the compressed data buffer end of the data link is a data link controller. The controller must send packets of data, taken from the compressed data buffer, to PVP at a fixed rate and must store packets received from PVP back in the buffer. In addition to managing packet exchange with the data link controller, PVP must complete the packetization of data for network transmission according to the Packet Voice Protocol. Finally, PVP serves as a network host exchanging outgoing and incoming communication packets with the network gateway, the Voice Funnel.

Progress Update. Work on the ISI Packet Video System is nearing completion. Two sets of virtually complete system hardware have been installed, one at ISI and the other at Lincoln Laboratory. Two-way simultaneous video connections have been established between the two sites and are now made on a fairly routine basis. Progress in achieving video with a higher update rate is proceeding along two fronts: utilization of higher data rate "Stream Mode" transmissions over the channel and attainment of significantly higher compression levels in processing the video signal.

The present level of video capability is based on having working video hardware and baseline image processing and data-transmission software. The hardware system includes:

- the Adage RDS-3000 commercial digital video system with digital camera input and display. The system is equipped with the powerful bit-slice microprocessor (BPS) to control video I/O and supervise all other video system tasks.
- the Video Engine, the ISI-designed-and-built parallel processor which performs all the intra-frame processing based on the technique of block-transform coding. It is controlled by the BPS microprocessor.
- the HDLC channel, a high-speed serial interface for I/O of compressed data between Video System memory and PVP. It is also controlled by the BPS microprocessor.

Baseline software includes:

- intra-frame processing which implements block-wise the Discrete Cosine Transform and scales, codes and packs the data, for transforming pictures to compressed form, and reverses the process to expand compressed data back to pictures.
- command blocks and data structures for the HDLC channel.
- BPS executive programs which control video I/O, from the camera and to the monitor, operation of the Video Engine for image processing of a single frame and control of the HDLC channel for I/O of compressed data.
- PVP software which implements the Packet Video Protocol, manages packet exchange between the Video System and the Voice Funnel and is currently capable of establishing 2-site connections over the network utilizing the Datagram Mode.

The current image-processing scheme can be described as a "fixed-frame-rate" scheme. It operates at a constant data rate: per-frame data compression is fixed at 16:1 and the frame update rate is limited by the channel capacity provided by Datagram transmission. Work is in progress on a "variable-frame-rate" scheme which also operates at a constant data rate but per-frame data compression can vary from 16:1 to 128:1 allowing the frame rate to vary from 3.75 to 30 frames per second, the maximum rate available from the camera. Implementation of this scheme involves development of the inter-frame differencing portion of the data-compression algorithm, to be performed by the BPS microprocessor, and extension of PVP software to utilize the Stream Mode for network data transmissions. The Stream Mode will accomodate a higher data-rate bit stream from the Video System than the Datagram Mode.

8.3.3.1 Video system hardware

With two working copies of the hardware installed at ISI and Lincoln Laboratory, the Video System hardware is virtually complete. Fig. 8-2 is a system block diagram. The Video System is connected via the HDLC channel to the Voice Funnel which serves as a gateway to the network node. The network node consists of the PSAT, the Earth Station Interface and the Earth Station. The HDLC channel exchanges packets between the Video System and a single processor in the multi-processor Butterfly Computer which runs the PVP program. PVP interfaces with the Voice Funnel program which runs on other processors of the Butterfly Computer.

The Video System itself consists of the Adage RDS-3000 system, controlled by the BPS microprocessor, augmented by the two-board Video Engine which includes the

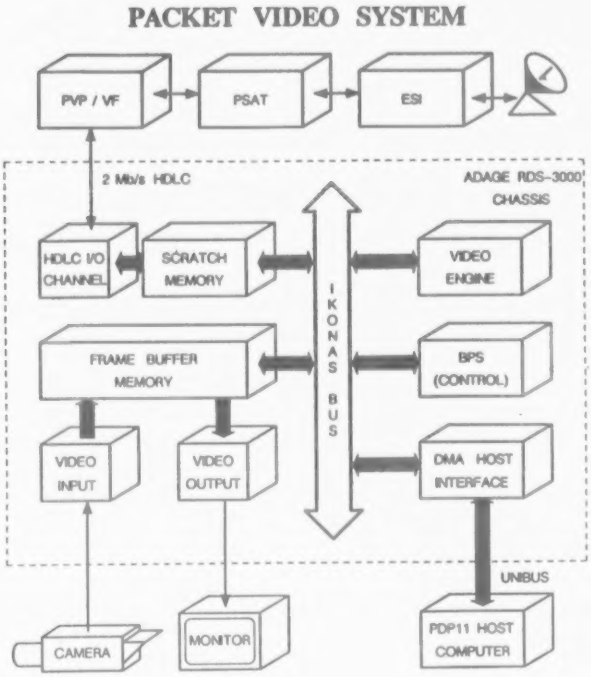
PACKET VIDEO SYSTEM
PVP / VF
PSAT
ESI
2 Mb/s HDLC
ADAGE RDS-3000 CHASSIS
HDLC I/O CHANNEL
SCRATCH MEMORY
VIDEO ENGINE
IKONAS BUS
FRAME BUFFER MEMORY
BPS (CONTROL)
VIDEO INPUT
VIDEO OUTPUT
DMA HOST INTERFACE
UNIBUS
CAMERA
MONITOR
PDP11 HOST COMPUTER

Figure 8-2: Video System Block Diagram

HDLC channel controller as a single Intel 82586 chip. The RDS-3000 system is a digital camera input and image display system which uses the Frame Buffer Memory to store digital images. The Video Engine performs image processing on a single frame. It can compress an image stored in the Frame Buffer Memory and store the result in the Scratch Memory for transmission to the network node via the HDLC channel. It can also expand a compressed image, received from the network node and stored in the Scratch Memory, and store the result in the Frame Buffer Memory for display. The BPS controls the Video Engine and the HDLC channel through a set of registers accessed over the Ikonas bus (fig. 8-2).

The Video System is hosted by a PDP 11/45 equipped with the EPOS operating system and a special-purpose Video System interface/debugger program, DUD, capable of independently running the BPS and Video Engine and loading and examining the various memories and registers.

Video Engine. The Video Engine is a special-purpose parallel processor designed to implement block-wise image processing algorithms. Fig. 8-3 shows a block diagram of the Video Engine. The Video Engine consists of two cards:

- a programmable Sequencer card, which controls external and internal I/O,
- a Processor card which contains 8 parallel microprocessors and their associated memories.

The Video Engine is controlled via a Command Register, a Source Frame Address Register and a Destination Frame Address Register. When commanded to run, the Video Engine inputs data from the Source Frame, processes it and outputs the new data to the Destination Frame.

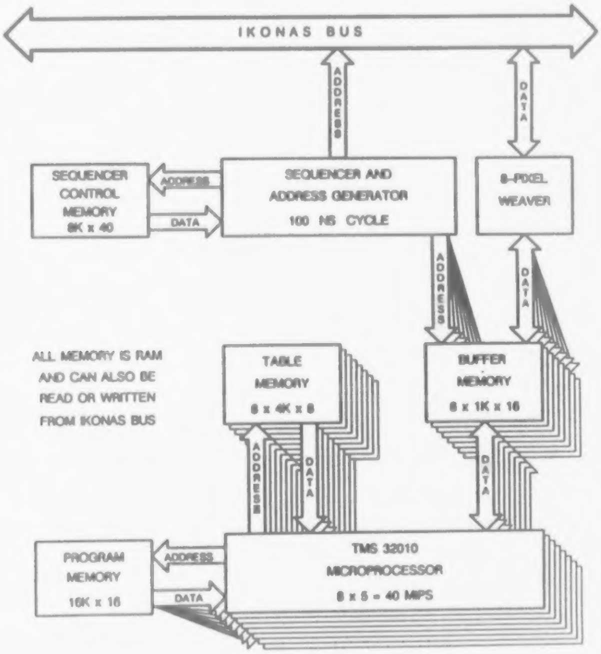
IKONAS BUS
ADDRESS
DATA
SEQUENCER CONTROL MEMORY 8K x 40
ADDRESS
DATA
SEQUENCER AND ADDRESS GENERATOR 100 NS CYCLE
8-PIXEL WEAVER
ADDRESS
DATA
ALL MEMORY IS RAM AND CAN ALSO BE READ OR WRITTEN FROM IKONAS BUS
TABLE MEMORY 8 x 4K x 8
BUFFER MEMORY 8 x 1K x 16
ADDRESS
DATA
DATA
PROGRAM MEMORY 16K x 16
ADDRESS
DATA
TMS 32010 MICROPROCESSOR 8 x 5 = 40 MIPS

Figure 8-3: Video Engine Block Diagram

The Sequencer performs external I/O using the Ikonas bus (fig. 8-3) between an external memory and one of eight internal 1K by 16-bit buffers, one for each of the eight parallel processors. Internal I/O is performed between the eight internal buffers and the corresponding parallel processors. External I/O involves only one buffer at a time whereas internal I/O involves all eight at once. The eight internal buffers, lying between the eight parallel processors and the two external memories, for image data and compressed data, allow the high speed (multiple store and fetch) I/O of the parallel processors to be interfaced with the lower speed (single store and fetch) I/O characteristics of the external memories. The Sequencer program is stored in an 8K by 40-bit RAM supplemented by a 4K by 20-bit RAM for the Block Address Memory.

Address generation for internal and external I/O is the core feature of Sequencer programmability. Internal addresses are wholly contained in sequencer program memory and external addresses are generated in a 3-stage fashion. The Sequencer instruction contains a pixel offset (within a block) which is added to the current block offset (within a frame), located in Block Address Memory, which is added to the source or destination frame address (see fig. 8-4). Programmability makes possible a variety of block structures and block decompositions of an image frame.

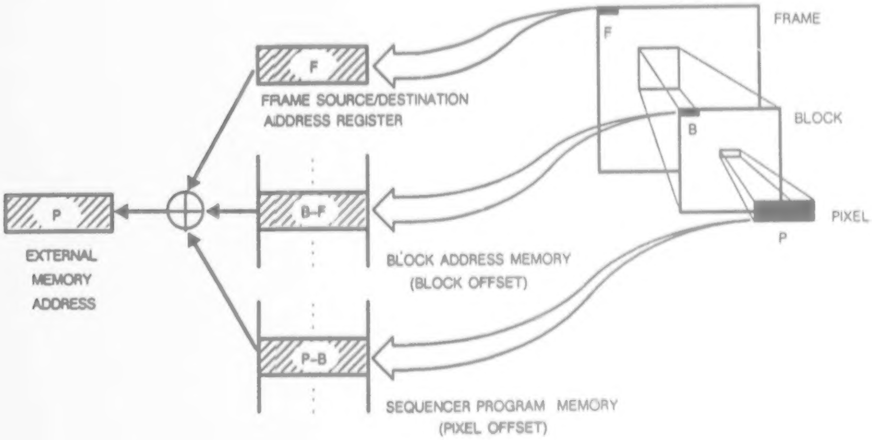

Figure 8-4: External Address Generation

The single-instruction-multiple-data parallel architecture is implemented using eight Texas Instruments TMS 32010 microprocessors (hereafter "320"). Each 320 is a 16-bit microprocessor equipped with a 16x16 multiplier, a 32-bit accumulator and 144-word internal RAM. Since each 320 runs at 5 MIPs the Video Engine has a peak speed of 40 MIPs. The 320s access data from their 1K buffers, process it and write the results back to the buffers. The Sequencer coordinates this operation by providing buffer addressing. Instructions accessed from a single shared program memory are executed in unison by all eight processors albeit on separate data. In this way eight buffers worth of data are processed simultaneously. While the eight 320s share the 16Kx16-bit program memory, each 320 has its own 4Kx8-bit table memory for table look-ups associated with the coding and decoding of transformed data.

HDLC. The final component of Video System hardware is the HDLC high-speed serial interface data link connecting the Video System and the PVP processor located in the multi-processor Butterfly Computer. The HDLC channel controller is implemented using the Intel 82586 co-processor which is capable of 2 Mbps half-duplex transmission.

The controller chip is capable of stand-alone operation and accesses programmable data structures and command blocks, co-located with data in Scratch Memory, through a shared port. The programmable data structures allow volatile (data) and static (header) portions of data packets to be separated in Scratch Memory and combined at transmit time. The Intel 82586 chip is physically located on the Sequencer card and controlled via the Video Engine Command Register.

8.3.3.2 Intra-frame coding

Before intra-frame coding takes place each 512x480-pixel camera image is subsampled to obtain a 256x240-pixel frame for image processing. Intra-frame coding implements a block-transform-coding algorithm on selected blocks of the given frame. The 256x240-pixel frame is decomposed into a 16x15 (row by column) array of 16x16-pixel blocks. Each block is processed by the algorithm in exactly the same way. To obtain compressed data from an image block, the algorithm proceeds through the steps of forward processing:

- (forward) Discrete Cosine Transform,
- scaling,
- coding,
- packing of the data.

To retrieve image data from a compressed block, inverse processing reverses the steps:

- unpacking,
- de-coding,
- de-scaling,
- (inverse) Discrete Cosine Transform.

All of these steps are performed by the TMS 32010.

Forward-transformed data must be scaled and coded to create compressed data which in turn must be de-coded and de-scaled before inverse processing is performed. In addition, routines have been written to pack the compressed data in a form suitable for network transmissions and to unpack it for inverse processing after it has been received. Because of the ample 16K TMS 32010 program memory, long versions of these programs have been created (which mostly replace software loops by straight-line code) which run up to 47% faster than the original versions.

By tuning the tables used for scaling, bit allocation and coding, excellent results have been obtained at the 2 bits per pixel (4:1 compression) coding rate for static images. Compression to below 2 bits per pixel results in reconstructed images which are less satisfactory. Starting from the 512x480 camera image the cumulative compression is 16:1 (4:1 from subsampling and 4:1 from intra-frame compression). All experimentation with intra-frame coding can now be done using the real hardware instead of software simulations.

8.3.3.3 BPS real-time executive programs

Presently the BPS implements a fixed-frame-rate image processing scheme capable of a maximum rate of 2.5 frames per second. The executive program cycles through the following tasks:

- input one video frame from the camera, 240 lines by 256 pixels,
- command the Video Engine to forward transform all 240 blocks of the input frame,
- receive from PVP, via the HDLC channel, 60 packets of compressed data received over the network (and 1 control packet),
- command the Video Engine to inverse transform all 240 blocks of the received compressed image for display,
- transmit to PVP, via the HDLC channel, 60 packets of compressed data produced by the forward transform and 1 control packet.

When fully optimized, the BPS executive program can complete this cycle in 1/4 second, resulting in a two-way transmission rate of 4 frames per second. Without optimization the expected rate reduces to 2.5 frames per second. Practical experience with the channel showed that rates of only about 1/2 to 1 frame per second could be achieved.

8.3.3.4 PVP software

PVP is a critical software element in the data transmission path. While it is part of the Video System software, it runs on a processor in the Butterfly Computer which is physically separate from the Video System and only connected via the HDLC data link. PVP has two I/O ports, one for the Video System and one for the Voice Funnel which serves as a gateway to the Network node. On the transmit path, PVP completes the packetization of video data according to the Packet Voice Protocol [3], performs packet bookkeeping chores and regulates the flow of packets to the Voice Funnel. On the receive path, PVP groups and orders all packets belonging to a single frame and detects and replaces missing packets with constant grey-level data. PVP does further consistency checks and packet bookkeeping and regulates the flow of packets to the Video System. PVP functions asynchronously with the Voice Funnel but synchronizes packet flow with the Video System, on a frame-by-frame basis, by exchanging control packets with the BPS. PVP also provides an operator interface for establishing network connections and provides statistical displays for operational monitoring.

8.3.3.5 Design of the inter-frame differencing algorithm

Most video compression systems utilize a frame-differencing algorithm to detect motion or the lack of motion. While appreciable motion generally behooves the system to operate at its maximum update rate, less motion may allow lower data rates. Lower data rates can be achieved by some combination of lower frame update rates or less data sent per frame.

The scheme being designed and implemented attempts to pick out the blocks in each frame which need updating and ignores the rest except for periodic refreshment. Once the update list has been constructed, it is placed in the Block Address Memory so that the forward transform will compress only the blocks chosen for update.

To construct the block update list, the algorithm compares each block in the new frame with the corresponding block in the previously processed frame. To determine whether the new block is sufficiently changed to require updating, values of selected pixels in the new and previous blocks are compared. The update decision depends on whether the number of pixels which crossed a difference threshold exceeds a certain count threshold. If not enough blocks of a given frame have changed to fill up the available processing time, then blocks will be added to the update list for refreshment, proceeding through the list of 240 blocks in a cyclic fashion.

Since the blocks processed and transmitted comprise only a partial list of all 240 blocks in a frame, information about the location of the blocks in the frame must be sent along with the compressed data. Hence the block update list must be extracted from the received data and written to the Block Address Memory before inverse processing is performed. In this way only those blocks singled out for update by the frame-differencing algorithm will be updated in the received picture.

8.3.3.6 Design of a BPS executive program for variable frame rate

In order to mesh inter-frame with intra-frame compression, and to meet various timing constraints, a variable-frame-rate scheme for BPS executive control is being designed and implemented. This scheme is designed to process all the blocks in each frame which are in the update list created by the frame-differencing algorithm (also executed by the BPS). Since the time to process a frame is proportional to the number of blocks to be updated, increased motion, which results in a longer block update list, is accomodated by a reduction in the frame rate.

To deal with the mismatch between frame update rates for the received and transmitted pictures, the processing of each image frame has been broken into fixed 32-block units. At the 4:1 compression rate implemented in the Video Engine, 32 blocks of forward and inverse processing is all that can be accomplished in 1/30 second, the maximum frame rate from the camera. Thus if fewer than 32 blocks change per frame, then the update frame rate can keep up with the camera (scanning at full rate) while sustaining a two-way network transmission. To handle the general case, for each new frame to be transmitted, the changed-block list is divided into 32-block sets, with the remainder filled in by refresh blocks. Each set is handled by the forward processing portion of a new unit processing cycle until the frame has been completed. Received images are likewise broken into 32-block sets which are processed in the inverse portion of each unit cycle, but update of the image display is held until all blocks of each frame have been processed. Each unit cycle thus handles 32 blocks of forward and 32 blocks of inverse processing.

By shuffling the 32-block units of forward and inverse processing, constant transmit and receive data rates are sustained while accomodating varying amounts of processing per frame. The data rate is fixed because unit processing cycles occur at a fixed rate and each cycle processes exactly the same amount of data. The frame update rate, however, can vary from 30 frames per second, when the update is completed in one processing cycle, to 3.75 frames per second, when 8 cycles are required to cover the 240 blocks of a full frame. The 32-block shuffling allows the transmit and receive frame update rates to vary independently.

The 32-block unit processing cycle also smooths out packet flow between the Video System and PVP, as compared to processing on a full-frame basis in the fixed-frame-rate scheme, and thereby reduces burstiness in PVP's own processing.

8.3.3.7 Expansion of PVP software

PVP must perform its tasks of packetizer and packet manager while meeting the I/O constraints imposed both by the Video System and by the Voice Funnel. Several modifications are currently underway.

To accomodate the needs of the variable-frame-rate scheme, PVP must handle more complex packet headers, which include the location of the packet data within a frame, and must handle a more intricate synchronization scheme for the exchange of packets with the Video System. On the other hand, while packets may still get lost in transmission, packet loss can be ignored by PVP and inverse video processing, just as though the lost blocks were not in the update list. Thus, ordering and replacement of missing packets are not required. (If fewer than 32 blocks are available for processing in any cycle, the BPS must make up the deficit by adding dummy addresses to make a full 32-block list for the Block Address Memory.)

To make available the more efficient Stream Mode for transmission over the channel, PVP must be able to manage Stream (ST) Protocol connections over the network, in concert with the Voice Funnel, and must tailor the packetization of video data to the ST protocol.

8.3.3.8 Real-time system milestones

The Video System is currently capable of fixed-frame-rate datagram transmissions. Current frame rates are low, not exceeding one frame per second. A number of distinct milestones were passed in the achievement of this level of operation:

- Nov. 2, 1984: First network transmission (a single static image was sent from Lincoln Laboratory and received and displayed by the ISI Video System).
- Jan. 11, 1985: Installation of a second copy of the Video System hardware at Lincoln.
- Jan. 18, 1985: Demonstration of two-way datagram transmission of fixed-frame-rate video between ISI and Lincoln.

In addition to the installation of the Sequencer card sets at both ISI and Lincoln, two-way video required the completion of several important software elements. The basic compression program which runs in the Sequencer had been extended to produce data formatted for transmission. A PVP program capable of fixed-frame-rate datagram transmissions had been produced and a BPS executive program for fixed-frame-rate processing was also complete.

8.4 IMPACT

The realization of a truly integrated data, voice, and video-communications network on the Wideband Network is approaching. Various possibilities exist for the separate and integrated use of these media in communications. In particular, the development of true multi-media communication systems, for conferencing or two-way communication, now seems to be a feasible goal. However, single-medium communication will continue to be a standard fixture in an integrated network.

The high-data-rate capability offered by the WBNET seems to be very attractive in the development of networks to share supercomputer facilities. Both the DARPA Strategic Computing Program and NSF are pursuing efforts to augment the potential of supercomputers with network access.

Packet Voice is expected to be an integral part of future Wideband Network communication. Single-medium use will most likely come first since the demonstration of high-volume voice communication is a priority. However, packet voice will surely figure prominently in experiments with multi-media communication packages.

While currently the least mature of communications media making use of the Wideband Network, packet video will also doubtless have a strong appeal to developers of multi-media communication packages. The likely progression in packet video will be from video-only to integrated voice-video and then to multi-media systems.

8.5 FUTURE WORK

ISI will continue efforts to maintain and improve Wideband Network system reliability and performance. Experiments with packet voice and packet video will be used to stress Network capabilities and yield areas for improvement and refinement. ISI will continue independent monitoring of the Network through its program of hourly testing and reporting of network connectivity.

8.5.1 Packet Voice

Commencement of large scale voice experiments can be expected in the near future. Following equipment upgrades for the Wideband Network and integration of the BBN STNIs into network Voice Funnels, this will be the primary objective of Packet Voice work. ISI will participate in the experiments necessary to bring high-volume voice communication to operational reality. The addition of the single-chip echo canceller to the STNI will now also be a priority.

8.5.2 Packet Video

The Wideband Project has succeeded in its initial efforts to develop a baseline Video System capable of transmissions over the network channel. Work is in progress to complete the implementation of the image processing and data-transmission design in order to utilize the full data-rate capacity of the channel, to achieve expected levels of video compression and to track motion in a visually satisfactory manner.

The following steps are anticipated:

- develop the capability for Stream Mode network transmissions,
- implement the frame-differencing algorithm,
- implement the variable-frame-rate image processing scheme,
- improve the fidelity of motion tracking,
- develop a data-prioritization scheme to insulate the Video System from degradations in WBNET channel performance.

Modification of PVP software to enable the use of ST connections will first make possible fixed-frame-rate, Stream Mode video between ISI and Lincoln Laboratory. This is expected to allow higher frame-rates. The next step is variable-frame-rate, Stream Mode transmissions. With the completion of the variable-frame-rate scheme, and using Stream Mode, appropriate tuning of the Video System and the network should result in variable-frame-rate performance which will reach the maximum 30 frames per second update rate.

Since the variable-frame-rate scheme does not increase the frame update rate when faced with increased motion, but slows down, an additional algorithm managing trade-offs between picture quality and motion tracking will be required. The effect of different compression levels, in the intra-frame compression function, on the trade-off between picture quality and motion tracking is not sufficiently understood. While the current implementation of the ISI Video System is not capable of dynamically changing the intra-frame compression levels, the effect of different levels and the possible advantages of dynamic changes in level can be explored. Rather than changing intra-frame compression levels, the more likely path to the solution will be dynamic variation of the frame-differencing algorithm. This will allow it to be less stringent in the face of increased motion, and not produce so many blocks for update, while being more stringent during reduced motion, thus still producing a reasonable number of blocks for update.

Since the packet-switching technology of the Wideband Network incorporates data-prioritization concepts in the handling of packets, appropriate prioritization of video data provides a method to compensate for degradations in channel performance, especially fluctuations in channel capacity. The nature of motion in moving video, and available choices involving picture resolution, suggest the possibility of data-prioritization which can lend itself to "graceful" degradation of the picture when some of the data must be discarded by the network [2].

Further operational experience with video communication over the network will accumulate as these final steps in the Video System development are taken. This experience will provide the basis for an overall evaluation of the suitability of the Wideband Packet Satellite Network for video communication.

REFERENCES

1. Bisbey, R. II, and D Hollingworth, *A Distributable, Display-Device-Independent Vector Graphics System for Command and Control*, USC/Information Sciences Institute, RR-80-87, 1980.
2. Casner, S.L., Cohen, D. and Cole, E.R., "Issues in Satellite Packet Video Communication," in *IEEE 1983 International Conference on Communications*, pp. A1.6.1-5, IEEE, June 1983. Also available as ISI/RS-83-5
3. Cole, E. R., *PVP - A Packet Video Protocol*, USC/Information Sciences Institute, W-Note 28, August 1981.
4. Forgie, J.W. and Weinstein, C.J., "Experience with Speech Communication in Packet Networks," *IEEE Journal on Selected Areas in Communications* SAC-1, (6), December 1983, 963-980.
5. Gitman, I., and Frank, H., "Economic Analysis of Integrated Voice and Data Networks: A Case Study," *Proceedings of the IEEE* 66, (11), November 1978, 1549-1570.
6. Roberts, L.G., "The Evolution of Packet Switching," *Proceedings of the IEEE 66* 66, (11), November 1978, 1307-1313.

9. KITSERV
VLSI KIT DESIGN SERVICE

Research Staff:	Support Staff:
Robert Parker	Jeff Deifek
Robert Hines	Shorty Garza
Jeff LaCoss	Jerry Wills
Richard Shiffman	

9.1 PROBLEM BEING SOLVED

Basic systems design philosophy is being changed by the growing ability of designers to submit prototype integrated circuit designs via computer network to an implementation service for relatively inexpensive, fast-turnaround packaged parts. The availability of such services allows products based on custom VLSI designs to become viable in low-volume and specialized marketplaces where traditionally high-volume, high-initial-cost VLSI devices would be prohibitive.

The need for design system support is growing with the availability of custom implementation services. Designers need tools to verify that their new devices perform intended logic functions.

9.2 GOALS AND APPROACH

The goal of the KITSERV project is to develop cost-effective test capability for VLSI designers as an integral part of their design environments, and to make this capability available to a large population of designers by transferring the technology to commercial companies willing to market and support the testers.

KITSERV will design and prototype a family of test environments to fill the designers' need to verify designs. These testers will cover a wide spectrum of capabilities, from simple stand-alone testers to complex systems operating in conjunction with VLSI design tools running on host machines.

KITSERV will initially focus on providing a test capability on the SUN workstation, which, along with the Berkeley VLSI design tools, will provide the designer with a relatively low-cost integrated design and test environment. KITSERV will encourage the migration of design tools to the personal computer level by providing a cost-effective test capability for these machines.

KITSERV will provide a planned comprehensive approach to developing tester products. As researchers will use DARPA-developed design tools on the SUN workstation, KITSERV will provide the ability to utilize DARPA-developed test capabilities as well.

KITSERV will also provide staged development through an initially conservative but timely offering followed by more agressive, higher utility product developments in several distinct phases. Kit 1, a packaged version of a test board developed at Stanford, will be available for demonstration during the CMOS classes to be held at ISI in August 1985. Kit 2, a KITSERV-designed higher performance vector tester for the SUN, will be available for demonstration in February 1986. Kit 3 will be a low-cost classroom device and will be demonstrated before the end of 1985. Kit 4 will be a second-year effort to design a highly integrated high-performance tester. Coincident with the development of these kits, KITSERV will initiate the licensing agreements to provide commercial sources of KITSERV products.

9.3 PROGRESS

As of June 1985, KITSERV is in a formative state. Early work on this project has been devoted to planning and specification of project goals, as outlined in Section 1.2. In addition, preliminary planning for Kit 1 has been performed.

Kit 1 will be a completely repackaged version of a Stanford-designed test board. The tester interacts with a host processor through a standard DMA channel and accepts commands to write individual pins or sense individual pins on the device under test (DUT). Kit 1 consists of five custom VLSI chips of two types on a custom printed circuit board. The functionality of the design has been extended by allowing devices with 28, 40, and 84 pins to be tested in addition to the 64-pin devices supported by the Stanford tester. Furthermore, connectors have been provided to allow Kit 1 to through-connect the host processor to a wafer probe station.

Since the Stanford tester has been demonstrated on a VAX via a DR11-W DMA interface supported by the SIEVE test language, and since there are DR11-compatible DMA devices for the SUN workstation, KITSERV will first attempt to install Kit 1 on the SUN using the Multibus version of the DR11-W. An appropriate driver for the SUN will be written or adapted, and the SIEVE language will be rewritten for the SUN.

Kit 1 will include an enclosure, a power supply, and a provision for a variety of DUT pin configurations. Building Kit 1 around the debugged Stanford chips gives KITSERV a package on which to design future products and allows the project to quickly proceed to its longer range goals of providing a higher performance test device as a second product. The Kit 1 tester will be packaged with power supplies and DUT adapters for the standard MOSIS-supported packages. (See Chapter 10). This first product will be demonstrated to the design community in August 1985. The repackaged tester will allow several problems to be solved in the early stages.

9.4 IMPACT

The KITSERV project is being initiated to develop test devices for the DARPA VLSI design community. By providing a design service responsive to the needs of the research community, the KITSERV project will significantly reduce the costs and delays involved in the testing and evaluation of new VLSI designs.

9.5 FUTURE WORK

In the next reporting period, KITSERV will design and produce several test environments integrated into the SUN workstation. Several production-ready prototypes of each kit will be distributed to the DARPA VLSI design community for evaluation and comments. KITSERV will then transfer the production and distribution responsibility to interested commercial companies under a non-exclusive license agreement.

9.5.1 Planned Activities and Products

Kit 1, described above, will be available for demonstration during the CMOS classes to be held at ISI in August 1985.

Kit 2, a KITSERV-designed higher performance vector tester for the SUN workstation, will be available for demonstration in February 1986. Kit 2 will be a higher performance tester than Kit 1. It will feature a microprocessor front end, a high-speed test-vector buffer on board, and more integrated test drive electronics. Furthermore, the addition of serial communications will make system interface more flexible than with Kit 1.

Kit 3 (ElfKit) will be a very low-cost physical environment for testing and evaluating student classroom projects. Serial and parallel data I/O will be supported to an extent consistent with complexity expected in a one-semester VLSI design project at both undergraduate and graduate levels. Prototypes of ElfKit will be demonstrated and distributed for comments in October 1985.

Kit 4 will be a high-performance tester. Beyond an intent to take full advantage of high-speed devices under development by DARPA VLSI contractors, no plans for this device have been finalized.

Preliminary specifications for Kit 4 will be developed and circulated as an RFC (Request for Comments) paper.

10. VLSI

Research Staff:	Support Staff:
George Lewicki	Barbara Brockschmidt
Ron Ayres	Kathie Fry
David Booth	Terri Lewis
Jeff Diefik	Lee Magnone
Joel Goldberg	Desaree Scott
Wes Hansford	Victoria Svoboda
David Hollenberg	Janna Tuckett
Shih-Lien Lu	Jasmin Witthoft
Lee Richardson	
Craig Rogers	
Barden Smith	
Vance Tyree	

10.1 PROBLEM BEING SOLVED

The VLSI design communities of DARPA and NSF require fabrication capabilities in order to investigate the design methodologies and architectures appropriate to VLSI where gate-count will exceed one million gates per device. Recognizing this need, DARPA established the MOSIS (MOS Implementation Service) system at ISI in January 1981. MOSIS has

- reduced the cost of VLSI prototyping;
- shortened turnaround time for VLSI prototyping;
- freed designers from fabrication idiosyncrasies; and
- made design less dependent on specific fabrication lines.

A cost reduction of one to two orders of magnitude has been achieved by spreading the fabrication cost over many projects. By centralizing (and computerizing) the idiosyncratic knowledge about all vendors, MOSIS eliminates the need for designers to familiarize themselves with many details. Serving as the only interface between its design community and the vendor base, MOSIS is able to provide turnaround times of four to six weeks for standard technology runs, except when unusual fabrication problems occur. Nonstandard technologies and experimental runs generally require longer fabrication schedules.

10.2 GOALS AND APPROACH

MOSIS involves various aspects of multiproject wafer assembly, quality control, and interaction with industry. The major components of the MOSIS system are

- interaction with the designers;
- handling of their design (CIF) files;
- communication over either the ARPANET or TeleMail;
- placement of projects on dies, and dies on wafers;
- matching of MOSIS design rules to specific vendors' design rules, addition of alignment marks, critical dimensions, and test devices;
- fabrication of E-beam mask sets (via subcontract);
- fabrication of wafer lots (via subcontract);
- wafer probing and data analysis;
- generation of bonding maps;
- wafer sawing, die packaging, and bonding (via subcontract);
- device distribution.

Designers use any available design tools to create artwork files, which are sent to MOSIS via the ARPANET or other computer networks. MOSIS compiles a multiproject wafer and contracts with the semiconductor industry for mask making, wafer fabrication, and packaging. MOSIS then delivers packaged IC devices to the user. The user perceives MOSIS as a black box that accepts artwork files electronically and responds with packaged IC devices.

Though MOSIS may be instrumental in providing cells and design tools to the user, it is the sole responsibility of the user to see that the submitted patterns yield working designs. One may compare MOSIS to a publisher of conference proceedings compiled from papers submitted in "camera-ready" form, where the publisher's responsibility is to produce the exact image on the right kind of paper using the appropriate ink and binding--but not to address the spelling, grammar, syntax, ideas, or concepts of the various papers.

MOSIS provides a clean separation of responsibility for the "printing" of chips. The semiconductor manufacturer is responsible for the processing of the parts and must satisfy MOSIS's rigorous quality control. MOSIS is responsible to the user for the quality and timeliness of the fabrication. The user is responsible for the proper design of the parts and may use any design methods he finds appropriate for his needs.

It is quite common that very advanced and sophisticated chips fabricated by MOSIS work on "first-silicon." An example of this is Caltech's MOSAIC--this is an amazing accomplishment of the existing design tools. Unfortunately, this is done at a considerable cost; for example, it is estimated that Caltech's MOSAIC chip consumed over 1,000 CPU hours on various VAXes before it was submitted to MOSIS for fabrication.

10.3 SCIENTIFIC PROGRESS

10.3.1 Technology Base for Fabrication Runs

nMOS

MOSIS routinely supports nMOS at 3.0 and 4.0 micron feature sizes, with *buried*, rather than *butting*, contacts, in accordance with the Mead-Conway design rules. At least 20 vendors can fabricate devices according to these nMOS design rules.

CMOS/Bulk

MOSIS supports two level metal both at 3 and 1.2 micron feature sizes.

CMOS/SOS

MOSIS supports CMOS/SOS fabrication with 4.0 micron feature size in accordance with the Caltech design rules. All of the SOS vendors support these design rules.

Completed Fabrication Runs

The following is a categoric breakdown by technology of the fabrication runs completed during this reporting period:

18	runs	nMOS, 3/4 microns
19	runs	CMOS/Bulk, 3 microns
2	runs	CMOS/SOS 4 microns

10.3.2 Computing

A substantial effort was made in this reporting period to transition the computing support for MOSIS from operating on two KLs to a number of clustered VAX-ll/750s. The transition is almost entirely complete; all that remains on the KLs is the MOSIS message service.

10.3.3 Fabrication Interface

The MOSIS vendor base has expanded substantially during this reporting period. Increased user feedback and more extensive test results have allowed the MOSIS project to determine and communicate fabrication requirements to new vendors. This has resulted in higher quality wafers and the development of consistently reliable vendor sources for mask making and nMOS fabrication.

MOSIS has instituted procedures to manage the vast amount of information inherent in dealing with a multi-vendor base. Many administrative tasks have been automated, including the maintenance of templates to determine fabrication requirements specific to vendor and technology (a single vendor often provides several fabrication technologies).

MOSIS now has a regular service on two-level metal CMOS/Bulk, with runs scheduled every three weeks. It has two sets of design rules. One is specific to three micron p well technology. The other is a scalable set applicable to 3, 2 and 1.2 p well and n well CMOS/Bulk technologies. MOSIS now offers fabrication at 1.2 microns. It is actively engaged in establishing a 2 micron vendor base.

10.3.4 Quality Assurance/Design Interface

Most MOSIS devices are prototypes without established functional testing procedures. Generally, the designers who receive these devices are still debugging the designs, rather than checking for fabrication defects introduced by less-than-perfect yield.

MOSIS's extensive quality assurance program is aimed primarily at the parametric level. This guarantees that the electrical properties of the wafers are within specifications established by the best a priori simulations used in the design process. Work has continued to increase the accuracy of the SPICE parameters which are made available to MOSIS users. SPICE provides simulated mathematical modes for behavior of transistors, allowing designers to assess a small digital circuit idea, to avoid faulty design, and to improve their chances of success in fabrication. The electrical criteria are a superset of the SPICE parameters at level II. They include a ring oscillator, which gives a rough idea of the speed of the resulting circuitry. The electrical properties of the wafers are extracted first by the fabricator, who uses either his own process control monitoring devices or the MOSIS test structures. Only wafers passing these tests are delivered to MOSIS.

It is a common practice in the IC industry to save functional probing time by probing wafers in only a very few sites. This practice makes sense, because all parts are subject to functional testing and because this parametric probing serves only to eliminate disastrously bad wafers.

Since designers hand-test most MOSIS devices, MOSIS requirements for parametric testing are higher than industry standards. MOSIS inserts its own test strip on every device, if space permits. When the test strips cannot be inserted, MOSIS probes a large number of test sites. This probing provides important statistics on the electrical properties and their distribution across the wafer. Most wafers have uniform distribution; some, however, have other statistical patterns, such as significant gradients and bimodal distributions.

These in-depth statistics are available only to the fabricators. Designers receive the general statistics (mean and variance) for each run. Interested users can request the specific values of the parameters extracted near any of their chips.

Users comparing the performance of actual chips to their simulations find it useful to rerun the simulation with the actual a posteriori parameter values extracted near that chip. The marking on each chip (which can be checked by opening the package lid and using a low-power microscope) identifies the run in which the chip was produced and its position on the wafer. Along with the wafer ID marked on the package, these markings provide the identification needed to determine the most relevant probing site.

A perfect set of electrical parameters does not guarantee perfect yield, so there is always a need for functional testing. MOSIS does not have the facilities for high-speed functional testing, but can perform partial functional testing. This screening typically catches the majority of fabrication defects, such as shorts. The screening is performed by applying and reading user-provided vectors to each device before the wafer is cut; those failing the test will not be packaged. By screening the larger chips--which typically have lower yield and higher packaging cost, and are required in larger quantities--MOSIS significantly reduces the packaging cost.

10.3.5 Standard Pad Frame/Packaging

MOSIS's current packaging strategy is to package enough parts to ensure a 90 percent probability of delivering a defectless part to the designer. This strategy was acceptable when most of MOSIS's community was designing small circuits and the fraction of packaged defective parts was small. However, a significant portion of the community has successfully completed the development of large designs and now wants from 300 to 3,000 working parts to begin developing prototype systems based on parts obtained through MOSIS. The yield for these large designs is expected to be 25 percent at best. If MOSIS were to follow its current strategy of packaging parts without any testing to indicate functionality, it would be packaging four times the required number of parts to achieve a requested quantity.

To avoid such waste, MOSIS has worked with Stanford University to define a functional test language (SIEVE) and has developed hardware to effect the testing specified by that language. Users now have the option of submitting text describing limited test procedures to be used at wafer probe to screen out bad parts. The purpose of this screening is to detect the types of "trivial" defects that cause the majority of bad parts and, therefore, to reduce packaging costs. Full functional testing is expected to be done by the user.

For designs with custom pad layouts, it is the responsibility of the designer to provide MOSIS with the custom probe card to probe his circuits. To eliminate the inconveniences associated with generating custom probe cards for every design, MOSIS has developed a set of standard pad frames, each specifying exactly where the pads are positioned. MOSIS stocks probe cards for each of the frames.

These standard frames are also expected to facilitate packaging. Bonding diagrams for projects currently submitted are generated manually, because several attempts to automate this process have met will only limited success. Bonding diagrams instruct the packager to connect a specific pad on the chip to a specific pin on the package. Standard pad frames have standard bonding diagrams, eliminating the the need to generate a new diagram for each project. The increased use of the standard pad frames has reduced the considerable amount of time needed to manually generate bonding diagrams. Standard frames also allow the bonding process itself to be automated. Automated, programmable bonding machines are currently available. Standard pad frames make possible a scenario in which an operator would identify a first pad and package pin; programmed information would then control the bonding on a chip.

Automating the packaging phase could significantly improve MOSIS's turnaround time. The best times experienced by MOSIS for various parts of a run in nMOS are (1) four days to convert submitted geometry into a format acceptable by a mask maker and to generate the first masks required by a fabricator to start a run; (2) five days for wafer fabrication; (3) one day for wafer acceptance; (4) seven days for packaging; and (5) one day for overnight delivery of packages. Standard pad frames with automated bonding could reduce packaging time to two or three days and reduce the time from submission of designs to mailing of packaged parts (in very conventional technologies) to approximately two weeks.

10.3.6 Standard Cells

MOSIS has acquired from DARPA a government designed standard cell library for 3 micron p well CMOS/Bulk, designed in rules identical to MOSIS and thus capable of being fabricated by MOSIS CMOS fabricators.

MOSIS is making this library available to its community. Moreover, MOSIS has contacted all the major commercial CAD vendors to induce them to support the library. A number of them have decided to do so. This allows the MOSIS community to purchase turnkey CAD systems to design standard cell based circuits which can be fabricated by MOSIS.

10.4 IMPACT

MOSIS's main function is to act as a single interface ("silicon broker") between a geographically distributed design community and a diverse semiconductor industry. As such an interface, MOSIS has significantly reduced the cost and time associated with prototyping custom chips.

The greatest impact of MOSIS, however, is in the community it has created. The MOSIS user community shares not only the fabrication services, but also experience, cells, tools, and software. The rapid growth of the community proves that the services provided by MOSIS are useful and important to both the academic and the R&D communities.

10.5 FUTURE WORK

With a large portion of the MOSIS community designing in CMOS, MOSIS will put considerable effort into providing better simulation models for CMOS devices so that there is a much closer correspondence between how a designer predicts his circuit to behave and how he observes it to behave after it has been fabbed through MOSIS.

11. ADVANCED VLSI

Research Staff:
George Lewicki
Ron Ayres
David Booth
Jeff Deifik
Joel Goldberg
Wes Hansford
Lee Richardson
Craig Rogers
Barden Smith
Vance Tyree

Research Assistants:
David Hollenberg
Shih-Lien Lu
Terri Lewis
Lee Magnone

Support Staff:
Barbara Brockschmidt
Kathie Fry
Desaree Scott
Victoria Svoboda
Janna Tuckett
Jasmin Witthoft

11.1 PROBLEMS BEING SOLVED

Military systems are currently limited, not by technology, but rather by the difficulty systems designers have in accessing that technology. In the past few years, the MOSIS project has successfully interfaced a large VLSI design research community to VLSI technology available within the American semiconductor industry. The result has been some remarkable innovations in computer architectures. The tools used were large computer resources and access to the community being served by a computer communications network.

Considerable innovation in systems designed for the Department of Defense is expected to occur if the approach used in MOSIS (interfacing a DoD VLSI design community to fabrication technology) is used to interface a DoD systems design community to the technologies on which it depends. The result will be a drastic shortening of the turnaround time required to produce prototype systems.

The main technology on which systems are dependent is VLSI. The natural first steps for this project are the provision of interfaces to 1.2 micron CMOS/Bulk technology necessary for systems based on 100,000-transistor chips; access to fabrication of printed circuit boards (PCBs) containing both custom-designed and standard VLSI parts; and access to centralized functional testing of custom-designed parts prior to their assembly into packages and boards. It is these services which must be provided for system designers if they are to easily assemble prototype systems.

There is normally a considerable time lag between the availability of a technology and a large systems design community being able to design systems which best take into account the characteristics of that technology. The availability of a technology simply means that processes have been developed which can reliably generate large numbers of devices on silicon. The time lag has to do with circuit designers being able to develop circuit design techniques and system designers being able to develop architectures that best take advantage of the characteristics of the technology.

CMOS/Bulk technology with a feature size of 1.2 microns is currently being developed in the research departments of a number of commercial organizations. It is not expected to be generally available until late 1985. In the normal course of events, one would not expect system designers to be able to take full advantage of the technology until about 1988. The work performed by the Advanced VLSI project is expected to shorten this normal course of events by a full three years. It allows the technology to develop in parallel with design rules and circuit and systems design techniques.

A VLSI system is a collection of integrated circuits electrically connected in a printed circuit board. Development of prototype VLSI systems requires access to a service generating PCBs from parts and a specification. Before PCBs can be fitted with parts, there must be some degree of assurance that the parts are functional. Economy dictates that functional testing begin at wafer probe, prior to the packaging of parts. The Advanced VLSI project provides DARPA systems designers with PCB sources and functional testing.

11.2 GOALS AND APPROACH

The primary goals of the Advanced VLSI project are to provide the DARPA VLSI systems design community access to 1.2 micron CMOS/Bulk technology and to PCB fabrication; and to provide centralized functional testing of custom-designed parts prior to their assembly into packages and boards.

11.2.1 CMOS/Bulk, 1.2 Micron

The 1.2 micron CMOS/Bulk was to have been conducted in three phases. The first phase of the project was to have involved a limited set of the DARPA research community submitting designs aimed at determining design rules applicable to a wide range of 1.2 micron bulk processes under development by U.S. industry. ISI was to carry out all testing of parts in-house, with results rather than parts supplied to the experimenters. The reason for this was security: the commercial organizations currently developing 1.2 micron CMOS/Bulk processes hold these processes as very proprietary, and they were willing to give the DARPA community early access to these processes only if security precautions were taken to prevent casual viewing of wafers. The only way in which such security could be guaranteed is to limit movement of wafers to the ISI environment. ISI developed a test specification language for experimenters to use to specify tests to be carried out on their designs. ISI was to these test specifications to carry out tests on devices generated within the first phase of this program.

During the second phase, MOSIS was to arrange fabrication runs for a selected segment of the DARPA research community. The primary purpose of these runs was to support circuit design experiments for the development of circuit building blocks. Again, all testing was to be performed at ISI and results were to be distributed to the investigators. ISI was to develop the necessary test languages and acquire the necessary test equipment to be able to carry out these measurements.

During the third stage of the project, when the technology had been fully developed, MOSIS was to offer fabrication in this technology at the rate of one run per month, to the whole of its community. During this phase, public viewing of the actual circuits would be considered acceptable, and parts would be assembled and distributed to the designers. At that time, measurements for parametric and yield test structures developed within the first phase of the project were to be used for the purpose of wafer acceptance.

The above constituted the original plan. Delays in fabrication and demand for the technology by MOSIS community caused phase 2 to be eliminated.

11.2.2 PCB Services

The purpose of the PCB service is to support the packaging requirements of the research community for assembling integrated circuits into systems. MOSIS supports the production of printed circuit boards in a manner similar to the ongoing VLSI activities, by interacting with designers (using the MOSIS net-based user interface), accepting designs (written in the language CIF), and converting them into working parts.

MOSIS maintains design rules and recommends procedures for PCBs, and provides instructions to users.

As many tooling formats are used throughout the industry, several software systems have to be developed to allow for the use of a wide vendor base. Further, in anticipation of problems arising in the move to finer geometries (such as 0.005 inches and 0.003 inches on a 24-inch board), MOSIS is developing software for driving laser plotters which play a role similar to E-beam mask making devices for VLSI.

11.3 PROGRESS

11.3.1 CMOS/Bulk, 1.2 Micron

MOSIS has successfully completed the first phase of its 1.2 micron CMOS/Bulk activity. It has obtained wafers from four potential 1.2 micron vendors, those wafers containing test structure the measurement of which allowed MOSIS to characterize the vendor's process.

The conclusion of this effort was a general process spec characterizing the range of processing that MOSIS would be able to access at 1.2 microns and a determination of the readiness of vendors to provide 1.2 micron processing. MOSIS is now distributing this general process specification to its user community.

MOSIS has at this time one vendor which it feels is capable of fabricating 1.2 micron circuits and has announced the availability of such processing to its community. It is certain that very soon, the other vendors it has been dealing with will be able to provide satisfactory processing.

11.3.2 Scalable Design Rules

1.2 micron technology is reticle rather than full wafer lithography based. This means that MOSIS will not be able to put tens of designs on one 1.2 micron run for prototyping purposes. With the cost of a run sharable among many designers, the cost of protytyping becomes very high.

MOSIS is taking the following approach to provide inexpensive prototyping to 1.2 micron designers. It is adopting a set of scalable CMOS rules. This means that designers can firstdebug parts of designs in cost sharable 3 micron CMOS runs and then do final development at 1.2 microns.

A set of scalable rules have been defined and distributed to the community.

11.3.3 PCB Services

MOSIS has been offering the fabrication of Printed Circuit Boards (PCBs) to its users. MOSIS treats PCBs as just another technology, surprisingly similar to nMOS and the various dialects of CMOS. Although PCBs and ICs are made of different materials, they share a common method of specification of the images required on each of their layers. Both are fabricated by a "photographic" process which transfers images to the media surface from a master tooling.

This approach of using common tooling preparation methodology for both ICs and PCBs has many advantages: it allows designers to use common tools for the design process (with details tailored specifically to each technology) and allows MOSIS to apply the same management procedures and the same geometrical processing and tooling preparation methods to PCB technology.

MOSIS has provided a total of 81 PCBs to nine designers since offering this service.

11.4 IMPACT

The Advanced VLSI project was initiated in order to interface the DoD systems design community to the technologies on which it depends. By providing access to fabrication of custom printed circuit boards, performing functional testing of parts prior to packaging, and working to make new VLSI technologies available to this design community, the Advanced VLSI project is significantly reducing the costs and delays involved in the design of prototype VLSI systems.

11.5 FUTURE WORK

11.5.1 CMOS/Bulk, 1.2 Micron

A number of fabricators are developing 2 micron CMOS/Bulk processes. 2 micron CMOS would be a better development background for 1.2 microns than 3 microns. MOSIS is now actively pursuing acquisition of a 2 micron vendor base.

11.5.2 Packaging

We will investigate more advanced packaging techniques, such as various ceramic carriers (direct inking, etc.) and plastic tape carriers. ISI is developing the necessary interface (software, frames, instructions for users, etc.) to support the use of these advanced packaging technologies by the community.

11.5.3 Other Services

As the needs of the systems design community become apparent, the Advanced VLSI project will move to meet them. Specific areas of interest in the coming year are the construction and maintenance of cell libraries and pad libraries, and a pad router that will support 1.2 micron CMOS/Bulk technology.

12. TEXT GENERATION FOR STRATEGIC COMPUTING

Research Staff:	**Research Assistants:**	**Support Staff:**
William Mann	Susanna Cumming	Lisa Trentham
Norman Sondheimer	Shari Naberschnig	
Mark Feber		
Christian Matthiessen		

12.1 PROBLEM BEING SOLVED

The US military is an information-rich computer intensive organization. It needs to have easy, understandable access to a wide variety of information. Currently, information is often in obscure computer notations that are only understood after extensive training and practice. Although easy discourse between the user and the machine is an important objective for any situation, this issue is especially critical in regards to automated decision aids such as expert-system-based battle management systems that have to carry on a dialog with a force commander. A commander can not afford to miss important information nor is it reasonable to expect force commanders to undergo highly specialized training to understand obscure computer dialects which change from machine to machine.

A great deal of work has been done in the area of natural language understanding and this work is starting to pay off with the delivery of functional interfaces that interact naturally with the user. Comparatively little, however, has been done in the area of natural language generation. There is currently no effective technology for expressing complex computer notations in ordinary English. If there were, computer-based military information could be made more accessible and understandable in a way which was less subject to personnel changes.

12.2 GOALS AND APPROACH

This project is creating and demonstrating new technology to provide an English-in, English-out interface to computer data, called **Janus**. ISI is developing the English-out (text generation) portion of Janus, referred to as **Penman**. Initial capability will be demonstrated on a naval database, but most of the techniques are more general and will be able to be reapplied to other military problems. The end result will be an exciting new capability for the Military that produces answers to queries and commands in the form of text that is understandable to any user that understands English.

Both understanding and generation components are necessary in an effective information system interface. It is also necessary that the technologies used by these two components be absolutely consistent. If they are not, the embarrassing situation of the interface failing to understand a piece of text which it has just output can occur, leading to a loss of confidence in the system on the part of a user.

In order to alleviate this problem, we are developing Penman in close cooperation with a natural language understanding project at Bolt, Beranek, and Newman Inc. (BBN). This cooperation extends to joint development of knowledge representation software, lexicon design, and work to insure compatibility between the RUS understanding grammar and the Nigel generation grammar.

The understanding/generation compatibility effort can be considered the first of our goals for the Penman system. Another is the ability to generate substantial amounts of English text, up to the multiparagraph level. A useful interface should not be restricted to short replies or comments.

A third goal is to make Penman as portable and domain independent as possible. There are many knowledge domains available for interface systems, and few specialists to create the interfaces. Thus, we are emphasizing domain independent capabilities over domain dependent ones.

Finally, there are many linguistically and computationally significant issues that must be addressed if high quality generation is to be achieved. For example, we are working on improved methods for knowledge representation, vocabulary development, and for using semantics and discourse context to guide the generation of text.

12.3 ACCOMPLISHMENTS

This project was put in place at the beginning of FY85 (October 1985) in order to develop the first natural language generation capability robust and capable enough to be used in DARPA's Strategic Computing Program. It is intended that this interface be coupled to the battle management system being developed under DARPA's Fleet Command Center Battle Management Program(FCCBMP). In the first nine months of the effort, USC/ISI was able to design the first generation system to produce English from output demands in formal mathematical logic using a broad coverage grammar and an artificial intelligence knowledge base.

As part of this design, the project has created and delivered a Master Lexicon facility, ML, for vocabulary acquisition and use [1, 2, 3]. ML is unusual in that it is compatible with the two radically different grammars of English in Janus: the Nigel generation grammar and the Rus understanding grammar. The system features a multi-window bit mapped display which is manipulatable through the keyboard and a mouse. Online documentation is provided for all lexical choices. ML is built to allow for the extension of the Janus lexical system. It is also possible to use ML with other natural language processing systems by replacement of a single data structure.

In addition, work has been done on bringing the Nigel and RUS grammars into compatibility.

We have as a general goal the development of natural language generation capabilities. Our vehicle for these capabilities will be a reusable module designed to meet all of a system's needs for generated sentences. The generation module must have an input notation in which demands for expression are represented. The notation should be of **general** applicability. For example, a good notation ought to be general useful in a reasoning system. The notation should have a **well-defined** semantics. In addition, the generator has to have some way of **interpreting** the demands. This interpretation has to be **efficient**.

In our research, we have chosen to use **formal logic** as a demand language. **Network knowledge-bases** will be used to define the domain of discourse in order to help the generator interpret the logical forms. And a **restricted, hybrid knowledge representation** will be utilized to analyze demands for expression using the knowledge base. We have developed a design that calls for:

1. Developing a demand language, Penman Logical Form (PLF), based on first order logic [8],
2. Structuring a NIKL (New Implementation of KL-ONE) network to reflect conceptual distinctions observed by functional systemic linguists.
3. Developing a method for translation of demands for expression into a propositional logic database,
4. Employing KL-TWO [9] to analyze the translated demands, and
5. Using the results of the analyses to provide directions to the Nigel English sentence generation system [7].

PLF is essentially complete. To first order logic, PLF adds restricted quantification, i.e., the ability to restrict the set quantified over. In addition, we allow for equality and some related quantifiers and operators, such as the quantifier for "there exists exactly one ..." ($\exists!$), the operator for "the one thing that ..." (ι). We permit the formation and manipulation of sets, including a predicate for set membership (**ELEMENT-OF**). And we have some quantifiers and operators based on Habel's η operator [4].

The system's knowledge has been organized in a new way in order to facilitate English expression. Abstract concepts corresponding to the major conceptual categories of English have been defined in NIKL (a KL-ONE dialect) and used to organize the conceptual hierarchy of the domain. By defining concepts such as process, event, quality, relationship and object in this way, fluent English expression is facilitated.

KL-TWO is a hybrid knowledge representation system that uses NIKL's formal semantics. KL-TWO links another reasoner, PENNI, to NIKL. For our purposes, PENNI, can be viewed as restricted to reasoning using propositional logic. We plan to translate a logical form into an equivalent KL-TWO structure. All predications appearing in the logical form wil be put into the PENNI database as assertions. A separate tree will be created which reflects the variable scoping. Separate scopes will be kept for the range restriction of a quantification and its predication.

The Nigel grammar and generator realizes the functional systemic framework at the level of sentence generation. Within this framework, language is viewed as offering a set of grammatical choices to its speakers. Speakers make their choices based on the information they wish to convey and the discourse context they find themselves in. Nigel captures the first of these notions by organizing minimal sets of choices into **systems**. The grammar is actually just a collection of these systems. The factors the speaker considers in evaluating his communicative goal are shown by questions called **inquiries** inside of the **chooser** that is associated with each system. A choice alternative in a system is chosen according to the responses to one or more of these inquiries. It is these inquiries which we will implemented.

Our planned implementation of Nigel's inquiries which will use the connection and scope structures with the NIKL upper structure is fairly straightforward to describe. Since the logical forms reflecting the world view are in the highest level of the NIKL model, the information decomposition inquiries will use these structures to do search and retrieval. With all of the predicates in the domain specializing concepts in the functional systemic level of the NIKL model, information characterization inquiries that consider aspects of the connection structure can test for the truth of appropriate PENNI propositions. The inquiries that relate to information presented in the quantification structure of the logical form will search the scope structure. Finally, to supply lexical entries, we will associate lexical entries with NIKL concepts as attached data and use the retrieval methods of PENNI and NIKL to retrieve the appropriate terms.

Besides the technology described above, work has been done on aspects of discourse modeling. This included a study of object description and an initial noun phrase planner.

In order to produce the sorts of larger texts that are needed in a User Assistance facility, work is under way to apply Rhetorical Structure Theory to text planning. A procedure has been developed which can design structures for paragraph-length texts given information about the text goal and communication situation.

12.4 FUTURE WORK

In the next phase of the work, we will turn our design into a working system.

Table 12-1 shows a series of independent sentences that illustrates the level of performance Penman is being developed for. Because of the current absence of a naval knowledge base, we have had to plan to test our ideas on the Consul knowledge domain of computer mail and calendars [5]. We have arranged and labeled the sentences to show how they might enter into computer database dialogue. Paraphrases in <brackets> show corresponding sentences of the sort that will be generated from Naval domain knowledge.

Penman will be delivered to BBN, integrated into Janus, and demonstrated in an extensive series of laboratory exercises. This version of Penman will support coordinated English input and output, paraphrasing, and user assistance, as well as constructive rhetorical structure theory to create descriptive texts [6].

For a machine to create fluent English texts, several problems of text organization and presentation must be solved. ISI will design, implement and demonstrate two kinds of new text planning processes. One will be a planner for gross organization of text based on rhetorical structure theory, and the other a low level planner for sentence

Table 12-1: Sentences from Penman

1. User: What is the read status of the message which was sent by Jones? <What is the readiness of the ship deployed by CINCPACFLT?>
2. System: Message 40 is unread. <The Nimitz is C4.>
3. User: Are any messages which were sent by Jones in the tape file? <Are any ships deployed by CINCPACFLT in the Atlantic fleet?>
4. User: Are any messages which were sent by Jones in the old ISI tape file which Smith has? <Are any ships deployed by CINCPACFLT in the new battle group which Spivak has?>
5. System: Yes. <Yes.>
6. System: Do you want me to display the read status of the messages? <Do you want me to display the readiness of the ships?>
7. User: No. <No.>
8. System: There are messages which were sent by Jones. <There are ships which were deployed by CINCPACFLT.>
9. System: However, the tape is in use. <However, the battle group is en route.>
10. User: Show me Message 40. <Show me the JFK.>
11. User: Did Smith send any messages? <Did the JFK send any CASREPs?>
12. System: Message 45 was sent by Smith. <CASREP 45 was sent by the JFK.>
13. User: Did Smith schedule his meeting with me in New York? <Did the JFK schedule its port visit in San Diego?>
14. System: Yes. <Yes.>
15. User: What does the window show? <What does the map show?>
16. System: The window shows a message which was sent by Jones. <The map shows a sail plan which was sent by CINCPAC>
17. System: The window shows a request by Jones. <The map shows a request which was sent by CINCPACFLT.>
18. User: Did I create the window? <Did I create the map?>
19. System: The window was created when you asked me to display Message 50. <The map was created when you asked me to display Sail Plan 33.>
20. System: You deleted the message, then you asked me to display it. <You deleted Sail Plan 33, then you asked me to display it.>

configurations. In addition, ISI will examine the expert system explanation functionality necessary within FCCBMP, and will design a text planner to meet those needs.

Enhancements to the basic interface to Penman will also be developed. As part of Janus, BBN plans to develop methods for translating English inputs into a higher order predicate calculus form. In addition the logical form will be manipulated inside the BBN understanding component to reflect database organization, expressions in this form are expected to be much more complex than the direct expressions of the corresponding English, requiring multiple sentence expression. Penman will be extended so that it can express these complex forms in English.

Following successful completion of laboratory testing, Penman will be delivered to NOSC for knowledge base development and testing. This Penman capability is a basic text generation capability with only limited capability for responding to the knowledge of the user and the state of the user-machine interaction. In subsequent years, these basic capabilities will be extensively expanded to provide a much more useful interface, with particular development in the areas of user assistance, ability of the system to understand the user's knowledge and expectations, and meeting needs for expert system explanation and user dialogue, as described below.

Development of extensions is required in the area of user assistance functionality. Text generation can assist the user by explaining system actions and objects, and by clarifying its interpretation of incoming English. These capabilities must be integrated into the Janus interface in a natural, easy-to-use form. We expect to demonstrate increased user assistance functionality as time goes on.

Development of extensions is also required in the area of responsiveness to users' knowledge and expectations. This includes making the system aware of the ongoing topics and issues, what the user has already been told, and what he can be expected to know without being told. This in turn involves more extensive use of inference. It also requires implementing a notion of the state of the dialogue -- what things are in attention, what interactions are in process and what has been interrupted and suspended.

Penman is scheduled for use in an expert-system explanation context whose details are yet to be determined. We will study what functionality is required in FCCBMP. The study will lead to a design of code for this use of Penman. Implementation and demonstration of full functionality are expected following the user assistance work.

REFERENCES

1. Susanna Cumming, *Design of a Master Lexicon*, USC/Information Sciences Institute, Marina del Rey,CA, Technical Report ISI/RR-85-163, February 1986.

2. Susanna Cumming, Robert Albano, *A Guide to Lexical Acquisition in the JANUS System*, USC/Information Sciences Institute, Marina del Rey,CA, Technical Report ISI/RR-85-162, February 1986.

3. Susanna Cumming, *The Lexicon in Text Generation*, USC/Information Sciences Institute, Marina del Rey,CA, Technical Report ISI/RR-86-168, 1986. Presented at The Workshop on Automating the Lexicon, Pisa, Italy, May, 1986.

4. Christopher Habel, "Referential nets with attributes," in Horecky (ed.), *Proc. COLING-82*, North-Holland, Amsterdam, 1982.

5. T. Kaczmarek, W. Mark, and N. Sondheimer, "The Consul/CUE Interface: An Integrated Interactive Environment," in *Proceedings of CHI '83 Human Factors in Computing Systems*, pp. 98-102, ACM, December 1983.

6. Mann, W., *Discourse Structures for Text Generation*, USC/Information Sciences Institute, Marina del Rey, CA, Technical Report RR-84-127, February 1984.

7. William C. Mann & Christian M.I.M. Matthiessen, *Nigel: A Systemic Grammar for Text Generation*, USC/Information Sciences Institute, Technical Report ISI/RR-83-105, Feb 1983.

8. *Penman's Logical Language*, USC/Information Sciences Institute, Marina del Rey, CA, 1985.

9. M. Vilain, "The Restricted Language Architecture of a Hybrid Representation System," in *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pp. 547-551, Los Angeles, CA, August 1985.

13. COMPUTER RESEARCH SUPPORT

Technical Staff:

Gary L. McGreal, Director

Systems Software:
Joel Goldberger
David Bilkis
Jeff Cavallaro
Dale Chase
Steve Ingersoll
Jim Koda
Mitchell Lerner
William Moore
Craig Ward
Tom Wisniewski

Hardware:
Daniel Pederson
Glen Gauthier
Ramon Gonzalez
Norman Jalbert
Raymond Mason
Jeff Rolinc
Daniel Trentham
Val Fucich
Christopher Welch
Mike Zonfrillo

Operations and Network Services:
Stephen Dougherty
Walt Edmison
Stuart Cracraft
Joe Kemp
Roger Lewis
Toby Stanley
Christine Tomovich
Roylene Trembley
Vicki Gordon
Sam Delatorre
Anna-Lena Neches
Wayne Tanner
Robert Wormuth

Support Staff:
Manon Levenberg
Nancy Garman
Pat Thompson

13.1 PROBLEMS BEING SOLVED

The Computer Research Support project is responsible for providing reliable computing facilities on a 24-hour, 7-day schedule to the ARPANET/MILNET research and development community. At the same time, the project makes available to ARPANET/MILNET users the latest upgrades and revisions of hardware and software. The project provides continuous computer center supervision and operation, and a full-time customer-service staff that is responsive to user inquiries. This project supports two computer installations, the larger at ISI's main facility in Marina del Rey. The Design Center at Gunter Air Force Base in Alabama is the other supported facility.

13.2 GOALS AND APPROACHES

The ISI Information Processing Center provides support in four tightly interrelated, though distinct, areas: Hardware, System Software, Operations, and Network Services. The overall charter of the organization is to assure that the needs of the user community be addressed and protected as efficaciously as possible. To achieve this end, each group is very concerned about the effective use of the machines as well as the security of the physical plant and the files and other online information. The more specific goals and approaches of each group are summarized below.

Hardware

To achieve a reliability goal of 98.7 percent scheduled uptime, the preventive and remedial maintenance responsibilities have been assigned to an in-house computer maintenance group. This group provides 20-hour, 5-day on-site coverage and on-call standby coverage for after hours. To maintain the reliability goals, preventive maintenance is very closely controlled, and on-line diagnostics and analysis are emphasized. A primary component in the reliability and availability of the hardware is the physical environment in which it exists. Accordingly, a significant amount of time and resources is expended in ensuring that the best, most cost-effective environmental controls are used at all facilities that ISI services.

System Software

The software group's primary goal is to install and maintain, at maximum reliability, ISI's VMS, UNIX and TOPS-20 operating systems and applications software. In order to accomplish this goal, the group provides 24-hour, 7-day coverage to analyze system crashes and to provide appropriate fixes. In addition, it is the group's responsibility to install, debug, and modify both the latest monitor and kernel versions and the associated subsystems available from the vendor.

Operations

The operations staff is responsible for operating the computers and watching over the systems and the environment. At the Marina del Rey facility there is 24-hour, 7-day on-site coverage. One of the primary responsibilities of the group is to provide protection and backup for user files to insure the integrity of the data. This goal is achieved through a variety of means, including regularly scheduled full and incremental backups of all systems; permanent archivals of requested or infrequently accessed system and user files; tape storage and pointers to all information extant at the time of removal of directories from the various systems; and, perhaps most important, redundant offsite storage of all significant information active on the disk structures or existing on tape within the facility. When a problem occurs, the on-duty staff isolates it and takes appropriate action. On the night and weekend shifts, the operations staff responds directly to user inquiries. Proper training, experience, and familiarity with the environment are especially important.

Network Services

Network Services, the ISI customer service group, provides a two-way communication link between the users and the entire support staff. This is accomplished by maintaining a 12-hour, 5-day on-duty staff for prompt problem resolution and rapid information exchange, both on-line and by telephone. The group offers introductory training in the use of both the hardware and software tools available on the ISI systems, as well as providing documentation for new users of the ARPANET/MILNET. Network Services also assists in the formulation of user training programs for large, INTERNET-based military experiments, such as, the U.S. Army XVIII Airborne Division Army Data Distribution System at Fort Bragg, North Carolina, and the C3/ARPANET Experiment at Headquarters, Strategic Air Command, at Offutt Air Force Base, Nebraska. Appropriate documentation is constantly being generated and distributed to individual users, as well as to remote user-group liaison personnel; this documentation ranges from simple, novice-level explanations to more technical information suitable for experienced users. In accordance with IPTO guidelines, the customer-service group provides regular accounting data to DARPA.

13.3 PROGRESS

The past year has been a very successful one for the Computer Center. Average uptimes were 99+ percent, and several major transitions have occurred without great difficulty, and more importantly, with virtual transparency to the user community. The most significant transition over the last year was the consolidation of all inhouse TOPS20 users onto one just one 2060 (while moving a substantial portion of their work onto other, more modern architectures).

Hardware Additions

Over the past three years the ISI Computer Center has undergone a major change in direction. Previously dominated by DEC PDP-10 computers, the facility was a relatively simple support environment consisting of a uniform collection of hardware and software. The current collection of equipment reflects the New Computing

Environment project plan of moving towards personal workstations and rear-end servers. The new collection of systems is a substantially more complex support problem involving three local networks (in addition to the ARPANET/MILNET connections), twenty different main processors (some of which have evolved into tightly clustered configurations with the acquisition in the last year of 2 HSC50 Cluster controllers acquired from DEC) and three major operating systems.

The Computer Center also houses and/or lends substantial support to the array of personal computers (PCs), workstations and symbolic processing engines acquired over the last several years on a variety of projects. The IPC assures that the filesystems of these various machines are backed up and that Network connectivity (where required) is assured in addition to physically housing major portions of most systems.

The current list of major multiuser processors, network servers and supported individual machines (PCs, workstations and symbolic processing computers) follows:

Quantity:	**Manufacturer:**	**Model:**	**Operating System:**
6	DEC	PDP-10	TOPS-20
1	DEC	VAX 8600	VMS/Ultrix
2	DEC	VAX 11/780	UNIX
1	DEC	VAX 11/780	VMS
2	DEC	VAX 11/750	UNIX
8	DEC	VAX 11/750	VMS
4	BBN	C-30 IMP	UNIX
12	Xerox	8010	STAR/MESA
3	Xerox	1100	Interlisp-based
3	3-Rivers	PERQ	PERQ
17	Symbolics	3600/3640/3645/3670	QLISP-Based
1	LMI	LAMBDA 2x2	QLISP-Based
2	TI	Explorer	QLISP-Based
10	SUN	100/120/150	UNIX
40	IBM	PC/XT/AT	MS-DOS

A map of the major pieces of the current local hardware configuration is shown in Figure 1-1.

System Software Enhancements

During FY'85 one of the major efforts centered upon making the above collection of hardware more compatible from a software point of view. Part of the work involved continued development and debugging of TCP/IP for the various operating systems, as

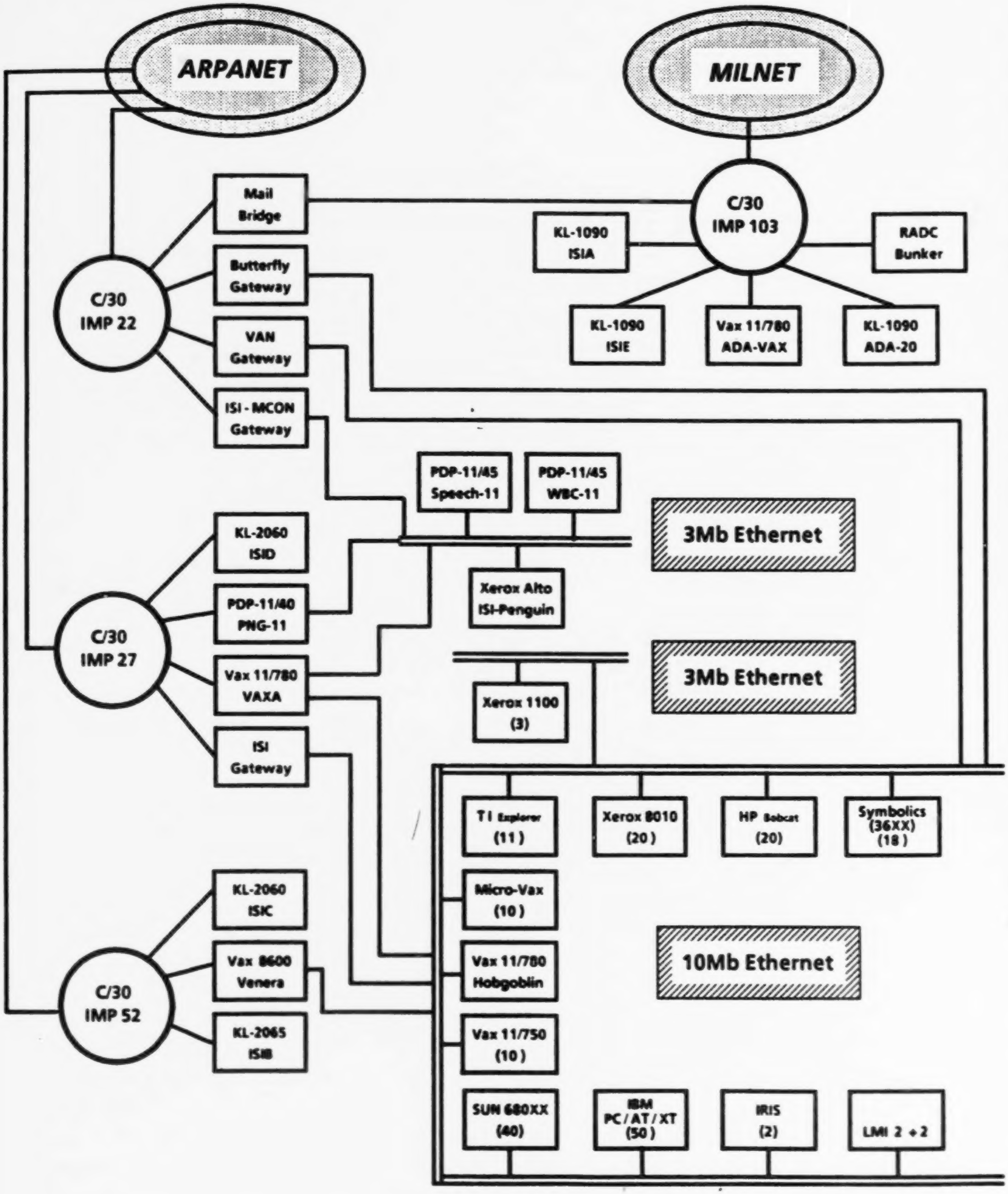
ARPANET
MILNET
C/30 IMP 103
KL-1090 ISIA
RADC Bunker
KL-1090 ISIE
Vax 11/780 ADA-VAX
KL-1090 ADA-20
Mail Bridge
Butterfly Gateway
C/30 IMP 22
VAN Gateway
ISI - MCON Gateway
PDP-11/45 Speech-11
PDP-11/45 WBC-11
3Mb Ethernet
KL-2060 ISID
PDP-11/40 PNG-11
Xerox Alto ISI-Penguin
C/30 IMP 27
Vax 11/780 VAXA
3Mb Ethernet
Xerox 1100 (3)
ISI Gateway
TI Explorer (11)
Xerox 8010 (20)
HP Bobcat (20)
Symbolics (36XX) (18)
Micro-Vax (10)
KL-2060 ISIC
Vax 11/780 Hobgoblin
10Mb Ethernet
Vax 8600 Venera
C/30 IMP 52
KL-2065 ISIB
Vax 11/750 (10)
SUN 680XX (40)
IBM PC / AT / XT (50)
IRIS (2)
LMI 2 + 2

Figure 13-1: Diagram of local ISI INTERNET facility

well as writing and installing new software for the hardware without TCP/IP orany other robust method of communicating with the other systems. One of the most notable of these efforts continued to be on the IBM-PC.

Another major effort involved the conversion or upgrade of major operating system releases themselves. The change with the largest impact on the user community was the upgrade of VMS to version 4 (in conjunction with the installation of the Clustering hardware).

Other significant development work over the last year included accounting systems enhancements on TOPS-20, upgrades to the accounting systems on UNIX and VMS and numerous bug fixes made on many System and user subsystem programs.

13.4 MILITARY IMPACT

ISI's computer centers provide ARPANET/MILNET cycles and support 24 hours a day, 7 days a week to the Strategic Air Command, Gunter Air Force Base and Fort Bragg as well as to the contractors, researchers and administrators of militarily significant research coordinated out of the DARPA office in Washington D.C. In addition to supplying machine time and directed Network Services support, this project continues to provide substantial additional support in the following areas:

- General accounting information on an as needed basis for workload analysis on the various machines.
- Rapid response to user requests and problem reports.
- Tailored security awareness and tracking both on a manual and on an automated basis.
- Maintenance and upgrading of electronic mail system software, shared online bulletin boards and other specialized communications and file transferral programs.

13.5 FUTURE WORK

The Computer Research Support project will continue to provide computing service to the DARPA research community, provide and support software packages for the ARPANET/MILNET community, and offer a program of technology transfer and user education through the Network Services group.

Some specific planned activities for the next year include:

- The final phasing over of local ISI staff to the NCE (New Computing Environment) resulting in the freeing up of the last TOPS-20 resources to the wider military community.
- Installation of new procedures and support for acquisitions of the NCE and other projects at ISI - including new products from Hewlett-Packard, Texas Instruments, SUN Microsystems, Symbolics, LMI and DEC.
- Installation of newer releases of VMS, UNIX and ULTRIX on our VAX computers.
- Installation of portions of TOPS-20 version 6.0 through 6.2 which will be of use to the ISI user communities.

14. EXPORTABLE WORKSTATION SYSTEMS

Research Staff:
Joel Goldberger
Gary McGreal
Jim Koda
Tom Wisniewski

Support Staff:
Manon Levenberg
Vivian York

14.1 PROBLEM BEING SOLVED

The use of dedicated, high-performance personal workstations and associated servers offer substantial advantages over terminals connected to time-sharing mainframes. The ever-increasing availability of integrated software environments has greatly facilitated the management and tracking of complex contracts, budgets for DARPA's program managers.

ISI provides the expertise necessary to configure the hardware and software of new workstations for the ISTO local area network and servers.

14.2 GOALS AND APPROACH

The primary goal of this project is to install and support a workstation environment at the DARPA-ISTO offices in Arlington, Virginia. It is also intended to provide a testbed for several other concepts and facilities. The environment is configured to accommodate a variety of workstation models with varying capabilities. This configuration provides a minimum set of services on each workstation model:

- document preparation,
- mail handling,
- file storage and manipulation,
- administrative activities (spreadsheets, calendars, etc.).

These services will be provided either directly on the workstations, or on the central server (a DEC VAX-11/750). All of the workstations will be connected to a local Ethernet and to the ARPA-Internet via a BBN-maintained gateway, and by equivalent code on the VAX that will function as a gateway in the event of failure.

To provide the DARPA-ISTO program managers with direct exposure to delivered software, we are working closely with the developers at other DARPA-sponsored centers to handle installation, customization, and problem resolution within the ISTO environment.

14.3 PROGRESS

Network disk service for the SUN workstations was established on a VAX/UNIX system at ISI. A number of changes were required to make it fully functional and reliable. Testing was done with the compatible workstations installed at ISI and further changes were made to improve the performance.

During this reporting period, the population of SUN workstations at DARPA-ISTO grew from 10 to 18. A VAX-11/750 was installed to function as a server for this population. The network disk service described above was brought up at the end of March. A number of third-party software products were obtained and installed to meet particular needs of the remote users. Sun Microsystems Inc. (SMI) released two new versions of the operating system which required minor fixes to both locally developed software and some of the imports. A utility was written and distributed to convert from the intermediate format of Interleaf documents to plain ASCII. A number of configuration issues were resolved to insure more reliable network connectivity and mail support.

A VAX-11/780 from MIT was installed to act as a backup machine to the VAX-11/750. This machine is also intended to run the MIT Planning System Software; however, this has not yet been demonstrated due to problems with the specialized display hardware.

Backup procedures were developed and installed to reliably protect the ISTO users against data loss. The current scheme involves backup to the VAXes located at DARPA; earlier schemes relied on machines at ISI.

14.4 IMPACT

The early acceptance of the workstations already in place and the ease of adaptation by previous time-sharing users clearly demonstrate the applicability of this new technology to office and administrative environments. This project has created considerable interest among our other user communities--military contractors in particular--as a means of providing increased computational capacity and enhanced functionality.

It is clear that dedicated personal workstations are superior to remote time-shared resources. The major advantages are greatly increased responsiveness of the system, elimination of delays caused by the Internet, and enhanced functionality available by virtue of the large format display and mouse pointing device offered on the workstation.

The ability to integrate additional workstation models into the environment and to have them function in an integrated manner will greatly enhance the portability and transfer potential of research efforts being carried out by most of the DARPA-sponsored groups around the country. The sponsors will be able to run the developed software on identical hardware without waiting for it to be adapted to another system.

14.5 FUTURE WORK

SMI has released their Network File System (NFS) software for both the SUN Workstation and VAX/UNIX. We plan to install this support both locally and in the ISTO environment. This software will allow sharing of files between the VAX servers and the workstations in a transparent manner and simplify software maintenance, installation, and backup procedures.

The ongoing support requirements of the ISTO environment are expected to increase with the installation of both our VAX-11/750 and the VAX-11/780 from MIT. We will develop procedures to provide reliable backup of both of these systems and attempt to utilize local contractors for onsite support as much as possible. We will also be developing a scheme to utilize the dual ported disk configuration to handle failures of either system and the multiple network connectivity of both systems to insure reliable access to the DoD Internet.

Penguin printer service at the ISTO office will be provided through these systems as well as print service for the remaining ISTO TOPS-20 users. This will allow us to decommission the PDP-11 currently serving this function.

DEC is expected to release a version of UNIX (ULTRIX) that includes better error logging and analysis facilities. We will either integrate these features into Berkeley 4.2 UNIX or integrate our local modifications into the DEC released product. Berkeley's next release (4.3) is also expected within this contract period.

Arrangements are in process to obtain and install several workstation software packages from other DARPA contractors (specifically BBN & SRI International) which may well require tuning and configuration for installation in the ISTO environment. The particular systems involved are the Multi-Media-Mail system from SRI developed under their contract with NAVELEX, and the Diamond mail system developed by BBN.

15. NEW COMPUTING ENVIRONMENT

Research Staff:
Joel Goldberger
Gary McGreal
Dale Chase
Jim Koda
Jon Postel

Support Staff:
Manon Levenberg

15.1 PROBLEM BEING SOLVED

ISI's two DEC mainframes running TOPS-20 are at full capacity and have become very costly to operate. Additionally, they are not capable of supporting many of the current and proposed research efforts at ISI. For the past decade, the computational needs of researchers in all areas of computer science were provided on large-scale timesharing machines. This is no longer a viable approach, and the availability of high-speed, dedicated workstations offer substantial advantages over timesharing.

As techniques and expectations (especially in the area of Artificial Intelligence) have advanced, general purpose machines are no longer able to provide the style of interaction and capabilities in terms of throughput and address space required to support current research efforts. The NCE project continues to:

- Provide a very significant improvement in both the amount and quality of available computing resources to the ongoing ISI research efforts through the use of dedicated personal workstations and higher capacity centralized processors and servers;
- Provide for diverse access methods to the common set of services;
- Fully support the utilization of special purpose workstations and processors as required by individual research projects;
- Free-up capacity on the existing mainframes by off-loading some common functions to central servers.

Due to the diversity of projects and cultures within the ISI research community, we have decided to implement a heterogeneous environment consisting of a variety of personal workstations, dedicated small mainframes, and continuing utilization of our existing timesharing systems. Initially, we will support this environment by means of autonomous facilities for file, mail, and document manipulation. This will be accomplished in a well coordinated way, as processes on a single server providing a unified view to the outside world. This is the most expeditious way of making the existing and yet to be acquired workstations fully functional. In parallel, we will begin an effort aimed at integrating these heterogeneous systems using the DARPA/DoD protocol suite (IP/TCP/FTP/...) and further provide mechanisms to promote the utilization of facilities among the various environments. All of the services and facilities we provide will eventually be accessible via the IP protocol suite and thus available from anywhere in the Internet.

15.2 GOALS AND APPROACH

The New Computing Environment is envisioned to consist of local processing nodes (workstations and multiuser mainframes) connected via a local network to a set of central servers. A certain minimum set of functions must be available to all nodes in this environment: mail, file manipulation (access and transfer), and Internet connectivity (Telnet). The environment provided by these nodes and servers should conceal the fine grain detail from outside users; users external to the environment need not know what workstation a particular user is on, unless they want to. At minimum, these services should be available via any of several communication media: Local Network, InterNet, and dialup access. The NCE will provide additional support as needed to allow lowend workstations and terminals on our mainframes to utilize these services as well as the fully capable workstations in the SUN, DLion, and Apollo class. Primary access to these services will be via the DoD IP/TCP protocol suite with some extensions and enhancements to support particular modes of interaction which are not yet supported (i.e., random file access and remote procedure call).

The utility of these services by our existing mainframes should realize immediate improvements for all users as some cpu intensive activities are off loaded to central servers. Making these services available from our mainframes will also provide for their use from home terminals, which initially will be glass teletypes rather than autonomous processors. Access for these devices will be via dialup lines to our existing mainframes, or to the servers themselves.

The services that provided to all nodes in the environment are:

- centralized mail service,
- centralized file service,
- document formatting,
- high-quality printing,
- communications, and

- (eventually), specialized processing.

While the principle incentive for the New Computing Environment is the support and integration of a diverse set of dedicated personal workstations, not everyone at ISI requires this kind of hardware and capabilities. Until they have an actual need to migrate their research efforts to workstations, there will be a significant population content with our existing mainframes (KLs and VAXes). However, the enhanced facilities offered by the provision of centralized servers should also benefit this population. These enhancements will take the form of increased accessibility of files and mail, increased reliability in terms of both data integrity and security, and improved performance of the existing mainframes due to the off-loading of CPU-intensive activities to the dedicated servers.

To reduce the operating and maintenance costs of in house computing, we have begun moving groups of users from TOPS-20 to VAXs running both UNIX and VMS. The TOPS-20 capacity freed by these transitions can be made available to other DARPA contractors, as determined by the program managers at IPTO.

15.3 PROGRESS

A survey was conducted of the in-house project leaders to determine their preferences and requirements for alternative computing environments to offload our existing DECSYSTEM-20's. Based on this survey and the available funding, we installed sufficient workstations, PCs, and increased VAX capacity to accommodate a subset of our local users. These acquisitions included 8 additional Xerox Dandelions and associated file server, 15 additional SUN workstations, 25 IBM-PCs, and added disk capacity on two of our existing VAX systems. Conditioned power was run to all of the offices in the institute to accommodate the distribution of this new equipment. We also completed the installation of a local area network throughout the 4 floors along with repeaters to facilitate fault isolation. Several additional laser printers were installed outside the computer room proper to allow easier access by the researchers.

To simplify support of this diverse workstation population we integrated all of the services they required on a single VAX 11/780 running Berkeley 4.2 UNIX. This includes the following servers:

- Chaos software for Symbolics 3600s,
- PUP/Leaf server for both Dolphins and Dandelions,

- Remote disk service for VAX 11/750s and IBM-PCs/XTs/ATs,
- Different disk server for SMI SUNs,
- Xerox Network System (XNS) server for Silicon Graphics IRISes.

Providing all of these services on a single machine facilitates backup and transfer of files among researchers working in different environments on different workstations. It also facilitates use of our printers and allows the exchange of software with other DARPA contractors via the DoD Internet.

Besides these VAX/UNIX-based servers we have also installed a dedicated disk server for the IBM-PC population to lessen the dependency on floppy disks and assure reliable and consistent backup. This server is also accessible over the Ethernet.

Mail support has been installed for most of these installed workstations. This support exists at various levels of completeness and utility. The SUN workstation comes with completely supported and functional mail software, though it functions contrary to our longer range design. The Symbolics machines make use of locally developed facilities to interact with the mail system on the VAX server. A new protocol was developed locally (Post Office Protocol - POP) to allow workstations to use either UNIX or TOPS-20 systems as mail servers. We have servers for this protocol running on both our DECSYSTEM-20 and VAX/UNIX systems. We have client implementations for the IBM-PC and SUN workstation. These client implementations are still being debugged and are not yet in daily use. A client implementation is being done for the Xerox Dandelion in the MESA environment built on our locally developed IP/TCP and is expected to be available for testing in May 1986.

15.4 IMPACT

Our experience with personal workstations, however limited, has made it clear that they represent a useful alternative to large, timeshared mainframes for certain research activities. Several projects within the institute have already moved the majority of their work to workstations, and have therefore been able to continue research which exceeded the capacity of our mainframes. The timely support provided for different workstation models has increased the acceptance of a distributed model of computation.

Once the reliability of added facilities and of the entire environment is proven, we will offer the same type of environment to outside contractors via the Exportable Workstation Systems project. These two projects combine to provide an ideal environment for developing and refining the facilities and capabilities that are becoming increasingly important in the command and control context. These capabilities include the handling of redundant databases and the interoperability of a diverse collection of hardware.

15.5 FUTURE WORK

We intend to continue the integration of services provided on behalf of our diverse workstation population. This includes more complete support for printing, file backup and archiving, and mail services. In particular we would like to utilize a single VAX as a mail server for all of the client workstations, rather than have them act as autonomous mail hosts as is presently the case. This applies to the SUN workstations in particular, though it will be expanded to include our VAX 11/750's as well.

Once these services are enhanced and proven reliable, our demands for TOPS-20 cycles will diminish significantly. Our goal is to reduce our eventual needs to approximately 25% of one machine.

Print service is currently being provided in a number of ways within the institute:

- Standard Line Printer service on our TOPS-20 systems.
- Penguin Laser Printer access via either TOPS-20 or directly from Dolphins, Dandelions and VAX/UNIX.
- Imagen Laser Printer access from all UNIX systems (VAX and SUN) and IBM-PCs.
- Diablo access via either standard terminal lines or from IBM-PCs via the 3Com server.
- Xerox 2700 Laser Printer access from all UNIX systems (VAX and SUN).

We intend to provide more consistent access to all of these printers and eliminate the present use of the Internet for moving files to specific printers. In particular this plan will eliminate the use of our current PDP-11 gateway to the 3MBit Ethernet for Penguin access from TOPS-20. This service will be provided by our VAX server. Facilities will also be developed to allow our TOPS-20 systems access to the Imagen and 2700 as backup for the Penguin.

We intend to explore the use of Laser Storage systems for long term archiving and possibly as a centralized public file storage system. These systems offer many advantages over our current tape backup techniques, mostly having to do with larger capacity and smaller physical size. Development of this facility is expected to require at least one full time programmer as well as some specialized hardware design. We envision this system as a network based service accessible to any client workstation or mainframe in the environment. We have already begun investigating the availability and capabilities of existing units and expect to decide on a prototype configuration very soon. The entire development effort is expected to take 9-12 months, with testing beginning as early as 6 months from date of initial acquisition.

Shared between the NCE project and the ISI Operation and Maintenance project will be continuing enhancements and fixes as needed for the tremendous body of software - both in house and planned acquisitions.

Security continues to be an important issue to our local users. Since the Ethernet is a broadcast network and because many of our monitoring and test facilities allow packet level monitoring there is concern that user interactions and data could be compromised. To address this we are investigating various public key encryption techniques that could be applied within our environment. This is an area in which there is active research being done by the Internet Research Group Task Forces and we will await results of that research for integration into our environment.

During the next year we expect to receive a number of other workstations for integration into the environment. These include the Explorer from Texas Instruments and several models of Hewlett-Packard workstations. It is expected that these will require the addition of special function servers on one or more of our VAX systems.

In a development effort shared with the Remote Monitoring and Maintenance project we will be developing a steady state Ethernet monitoring facility on an IBM-PC. This facility will help to identify network related problems. We expect to keep statistics on network utilization and eventually install capabilities to probe each node on the network to monitor its availability.

16. STRATEGIC COMPUTING - DEVELOPMENT SYSTEMS

Research Staff:
Gary McGreal
Dan Pederson

16.1 PROBLEM BEING SOLVED

Utilizing recent advances in artificial intelligence, computer science, and microelectronics the Defense Advanced Research projects Agency (DARPA) plans to create a new generation of "machine intelligence technology". The DARPA Strategic Computing program will be targeting key areas where advances can be leveraged towards broad advances in machine intelligence technology and demonstration of applications of the technology to critical problems in defense.

The Strategic Computing program will be supported by a technology infrastructure. This infrastructure will serve to bootstrap the program by providing current state-of-the-art computing technology, network communications, and shared resources to the selected program participants.

The aim of this project is to provide development computers for the Strategic Computing program, to provide system integration as required, distribution mechanisms for disseminating the systems to the program participants, and to define an architecture for system communications and resource sharing amongst and between the computers acquired.

16.2 GOALS AND APPROACH

A number of machines have been developed to support high speed processing of large symbolic programs. Each of these machines provide an extensive interactive programming environment, a sophisticated display manager "window system", a real time window-oriented editor, incremental compilers, and dynamic linking.

These systems are the state of the art in program development environments. They are widely used in the research community for systems development, expert systems, natural language systems, mapping applications and for supporting CAD/CAM environments.

There are several problems associated with utilizing these systems as a result of their high individual costs. Since they are single user systems they can not be timeshared in the traditional sense. The machines are too expensive to be supplied on a 1 to 1 basis to researchers, thus workers are currently placed in the awkward situation of having to schedule their computing needs. The resultant scheduling conflicts naturally lead to low researcher productivity.

An examination of the dynamic machine utilization shows that many activities that are carried out do not require the high speed processing capabilities afforded by these machines. Slower, less expensive general purpose machines supporting the same language base are becoming available. These would be adequate for many of the less intensive research activities such as editing, text preparation, and file scanning. They are too slow however for more intensive program development and execution activities.

The parallel duality in resources required for different types of activities and the availability of machines of appropriate cost/performance for carrying out those activities suggests a workstation/server architecture based on these two classes of machines and an interconnecting network. In order to support dynamic source code exchange between the server and workstation it will be necessary for each to support the same AI language system. Common Lisp is the natural choice in such an architecture as it was designed with the concepts of commonality and portability as primary goals.

A significant fallout of such an architecture, when viewed as workstations and servers on IP/TCP based networks, is the ability of researchers to communicate over the internet to utilize high powered resources available on prototype and low production machines. This will be particularly useful as prototype machines are developed under the machine architectures phase of the program.

ISI proposes to acquire a mix of these machines to support the requirements of the Strategic Computing program, and to integrate and test individual systems as required for particular Strategic Computing program participants.

The integration tasks will avoid duplication of effort amongst Strategic Computing developers. ISI will collect software to support this architecture as it is developed by vendors and the research community. ISI will test, modify, and augment software as required, provide support for software exchange and distribution amongst DARPA development sites, assure working compatibility of the Common LISP systems, work with commercial vendors to resolve vendor specific problems, and work to allow network compatibility of the systems with DoD network protocols (IP/TCP). The software and system integration efforts will be carried out at the ISI Marina del Rey facility.

16.3 PROGRESS

During FY'85, additional symbolic processing configurations were acquired. Some of them were installed and integrated for several months at ISI while most were shipped out to a number of research facilities, educational sites and government agencies.

In the past year there were 35 machines acquired as part of this Strategic Computing project. These machines were:

- 20 - Symbolics - 3640/3670
- 10 - Texas Instruments - Explorer
- 5 - Xerox - Danditiger

There are several of these machine in use at ISI in preliminary work on natural language, and the Common Lisp framework effort.

16.4 IMPACT

The DARPA Strategic Computing program will develop and integrate advanced computer technology into military applications. Technological advancement will be sought in the areas of microelectronics and high speed symbolic machines as well as application of this technology to the military.

Potential military applications of this technology include autonomous systems, battlefield management and assessment, planning and advising, and simulation systems.

The initial program applications included an autonomous vehicle, a pilot's associate and a carrier battle group battle management system. These applications test the evolving technology in the areas of vision, expert systems, speech recognition, and high performance knowledge processing systems. Each application seeks to demonstrate the new technologies' potential for providing a major increase in defense capability and to reflect a broad demonstration base that is relevant to each of the three services.

The development systems acquired will support the technology base and the targeted applications under the Strategic Computing program.

16.5 FUTURE WORK

Several corporate system integrators and chip manufacturers have perceived that there will be a substantial marketplace for symbolic processing engines in the future. It is our expectation that the price/performance ratio of these new machines will continue to improve at a rapid pace - resulting in more efficient utilization of quality researcher time as more state-of- the-art machines can be acquired at the most reasonable price.

During FY'86 ISI will continue to negotiate with qualified vendors and acquire a mix of high-end Lisp machines and workstations. As the workstation/server architecture is defined, ISI will configure systems, test vendor software, and distribute systems to Strategic Computing program participants as required and directed by DARPA.

17. STRATEGIC C3 SYSTEM EXPERIMENT SUPPORT

Research Staff:
Stephen Dougherty
Victor Ramos

17.1 PROBLEM BEING SOLVED

DARPA has defined an experiment in Strategic C3 systems to be conducted in cooperation with the World Wide Military Command Control System (WWMCCS) System Engineer (WSE) and the Strategic Air Command (SAC). The concept of the experiment is to demonstrate and evaluate the use of new technologies (such as the ARPANET, packet radio, network security, and distributed knowledge-base techniques) for strategic command, control, and communication.

ISI's portion of the plan is to provide an initial core of necessary facilities (ARPANET/MILNET access, host systems, various software tools, Network Services support, etc.) to allow SAC personnel to gain experience with this technology and to ensure the success of the experiment. Specifically, SAC must have fairly broad-based experience with ARPANET-based on-line interactive computing. Installation and maintenance of modems and of 55 or more interactive CRT terminals, user training, system software training and support, and on-site maintenance of equipment for the 300+ users at SAC Headquarters at Offutt Field, Omaha, Nebraska and at a handful of distributed SAC bases, are part of the continuing program.

17.2 GOALS AND APPROACH

The total specific goals of this DARPA experiment are to:

- Demonstrate and evaluate the survivability of multinode computer-communication networks, including the ARPANET and packet radio, especially for remote access from both airborne platforms and surface sites.
- Explore replication and reconstitution of critical knowledge bases on this network in the face of a loss of a large number of links.
- Demonstrate and evaluate the rapid reconstitution of a network by rapid deployment of packet radio nets to reestablish connectivity between surviving elements of the network.
- Conduct experiments and exercises to evaluate the utility of such a network on a distributed knowledge base to support postattack C3 activities.

The experiment is defined as having three phases:

1. Phase I is planned to demonstrate air-to-surface packet radio links and gateways into the ARPANET/MILNET as a first step in evaluating the feasibility of a truly survivable strategic network.
2. Phase II is directed toward creating a survivable message system and data bases through multiple copies of the critical components and data across the ARPANET/MILNET.
3. Phase III will address the feasibility of rapid reconstitution of a strategic network by deployment of packet radio networks to reconnect surviving elements of the network.

17.3 PROGRESS

The SAC user community has been growing since its inception. As a result, it has been a challenge to provide sufficient resources as the demand has spiked upward. An additional Terminal Access Controller (TAC) was installed to support the growing user community, additional lines and modems were also added to the Datacommunications base at Offutt Air Force Base. The dedication of ISIE to SAC helped eliminate user frustration and increase productivity due to the additional computational capacity available. ISI installed and now maintains terminals and communication equipment for the access of various Air Force bases to the MILNET which allows more widespread participation in the experiment.

During this period new resources were also obtained or redistributed to support the VAX van effort (the effort is a portion of the experiment testing the mobility and reconstitutability of a partially destroyed network). Additional ARPANET/MILNET access was provided to SRI International (which was working the Telecommunications portion of this effort and housed the vans) via 9600 bps communications lines.

17.4 IMPACT

One of the premises of this experiment is that SAC personnel will become more proficient and knowledgeable users of the current computer technology, available within the ARPANET/MILNET arena. This knowledge will allow SAC to make more effective evaluations of new technologies for strategic use, to identify areas of potential future research and to implement those technologies found appropriate.

17.5 FUTURE WORK

ISI will continue to assist DARPA in planning this program, working to satisfy the communication and computer resource requirements of SAC headquarters. In particular, ISI will do the following:

- Continue to provide on-site maintenance support for the required equipment.
- Continue to plan and assist in implementing improved SAC connectivity to the MILNET/DDN/ARPANET.
- Maintain terminals and communication equipment for the connection of several Air Force bases to the MILNET allowing increased participation in the experiment.
- Continue to supply computational, programming, and administrative support to users at SAC headquarters via the resources of the USC-ISIE computer, the programming, operations, and Network Services staff in Marina Del Rey, and the onsite technician.

ISI's most visible direct support will continue to be the on-site technician at SAC, who will be responsible for the identification of system malfunctions and for primary maintenance of on-site equipment. He will be supplied with required spare parts and will have maintenance contracts with the equipment vendors. Further support will be available from ISI in terms of additional spare parts, systems expertise, and documentation as necessary. The on-site maintenance technician will also be responsible for the off-site terminals at Vandenberg Air Force Base, Barksdale Air Force Base, March Air Force Base, and other locations as dictated by the requirements of the experiment. The on-site technician will coordinate data communications requests of SAC AD with SRI International and SAC under the supervision of ISI management. The technician will provide training and consulting to SAC personnel with backup from ISI as required.

ISI will provide program planning assistance to DARPA. We will continue to investigate the data communications requirements for SAC Headquarters, the HERT effort, and the ACCESS system (SAC Automated Command and Control Executive Support System). As specific research tasks are defined, ISI may choose to submit new proposals to address one or more of these tasks.

18. PUBLICATIONS

Books, Chapters, and Journal Articles

1. Cohen, Danny, and K. Fry, "PCB artwork preparation using E-beam pattern generators," *VLSI Design* 6, (8), August 1985, 82-88. Also published in *Proceedings of the IPC 28th Annual Meeting*, New Orleans, April 1985.
2. Leiner, B., R. Cole, J. Postel, and D. Mills, "The DARPA Internet protocol suite," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 28-36, IEEE, Washington, D. C., March 1985. Also in *IEEE Communications*, 23, (3), March 1985, 29-34. Also available as USC/Information Sciences Institute, RS-85-153, April 1985.
3. Mann, W. C., "The anatomy of a systemic choice," *Discourse Processes* 8, (1), January-March 1985, 53-74.
4. Postel, J., C. Sunshine, and Danny Cohen, "The ARPA Internet Protocol," in S. Lam (ed.), *Tutorial: Principles of Communication and Networking Protocols*, pp. 397-407, IEEE Computer Society, 1984. Reprinted from *Computer Networks*, (5), 4, July 1981.
5. Postel, J., "Internetwork protocol approaches," in W. Stallings (ed.), *Tutorial: Computer Communications: Architectures, Protocols, and Standards*, pp. 223-230, IEEE Computer Society, 1985. Reprinted from *IEEE Transactions on Communication*, (COM-28), 4, April 1980.
6. Shiffman, R. R., and R. H. Parker, "An electrophoretic image display with internal NMOS address logic and display drivers," *Proceedings of the SID* 25, (2), 1984, 105-115. Also available as USC/Information Sciences Institute, RS-84-144, March 1985.
7. Sproull, R. F., and Danny Cohen, "High-level protocols," in W. Stallings (ed.), *Tutorial: Computer Communications: Architectures, Protocols, and Standards*, pp. 405-420, IEEE Computer Society, 1985. Reprinted from *Proceedings of the IEEE* (66), 11, November 1978, 1371-1385.
8. Sunshine, C., "Formal techniques for protocol specification and verification," in S. Lam (ed.), *Tutorial: Principles of Communication and Networking Protocols*, pp. 467-473, IEEE Computer Society, 1984. Reprinted from *IEEE Computer*, (12), 9, September 1979.
9. Sunshine, C., "Transport protocols for computer networks," in W. Stallings (ed.), *Tutorial: Computer Communications: Architectures, Protocols, and Standards*, pp. 345-387, IEEE Computer Society, 1985. Reprinted from F. Kuo (ed.), *Protocols and Techniques for Data Communication Networks*, Prentice Hall, 1981.

10. Swartout, W., "Explaining and justifying expert consulting programs," in W. Clancey and E. Shortliffe (eds.), *Readings in Medical Artificial Intelligence: The First Decade*, pp. 382-398, Addison-Wesley, 1984.

11. Wile, D. S., "Transformation-based software development," in P. Pepper (ed.), *NATO ASI Series F: Computer and Systems Sciences*. Volume 8: *Program Transformation and Programming Environments: Report on a Workshop Directed by F. L. Bauer and H. Remus*, pp. 323-330, Springer-Verlag, 1984.

Refereed Conference Proceedings and Papers

1. Bates, R. L., D. Dyer, and M. Feber, "Recent developments in ISI-Interlisp," in *Conference Record of the 1984 ACM Symposium on LISP and Functional Programming*, Association for Computing Machinery, Austin, August 1984. Also available as USC/Information Sciences Institute, RS-84-131, August 1984.

2. Cohen, Danny, "The MOSIS story," in *The 4th Jerusalem Conference on Information Technology*, pp. 650-657, IEEE, Jerusalem, May 1984. Also available as USC/Information Sciences Institute, RS-84-139, July 1984.

3. Cohen, Danny, "Satellite communication of real-time packet video images," in *Proceedings of the Seventh International Conference on Computer Communication*, pp. 539-547, International Council for Computer Communications, Sydney, Australia, October 1984. Also available as USC/Information Sciences Institute, RS-84-136, June 1984.

4. Cohen, Donald, "A forward inference engine to aid in understanding specifications," in *Proceedings of the National Conference on Artificial Intelligence*, pp. 56-60, AAAI, Austin, August 1984. Also available as USC/Information Sciences Institute, RS-84-135, August 1984.

5. Kirton, P., "Some OSI experience of the ARPA-Internet," in *Proceedings of the Second International Conference on Introduction of Open Systems Interconnection Standards*, Ottawa, Canada, May 1984. Also available as USC/Information Sciences Institute, RS-84-142, July 1984.

6. Lewicki, G., "Prototyping and small-volume parts through MOSIS," in *GOMAC-85 Digest*, Proceedings of the Government Microcircuit Applications Conference, Orlando, Fla., November 1985. Also available as USC/Information Sciences Institute, RS-85-160, November 1985.

7. Mann, W. C., "Discourse structures for text generation," in *Proceedings of Coling84: 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pp. 367-375, ACL, Stanford, July 1984. Also available as USC/Information Sciences Institute, RR-84-127, February 1984.

8. Mann, W. C., and S. A. Thompson, "Assertions from discourse structure," in *Proceedings of the Eleventh Annual Meeting of the Berkeley Linguistics Society*, pp. 245-258, BLS, Berkeley, February 1985. Also available as USC/Information Sciences Institute, RS-85-155, April 1985.

9. Mockapetris, P., J. Postel, and P. Kirton, "Name server design for distributed systems," in *Proceedings of the Seventh International Conference on Computer Communication*, ICCC, Sidney, Australia, October 1984. Also available as USC/Information Sciences Institute, RS-84-132, June 1984.

10. Mockapetris, P., and J. Postel, "A perspective on name system design," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 349-355, IEEE, Washington, D. C., March 1985. Also available as USC/Information Sciences Institute, RS-85-152, April 1985.

11. Morgenstern, M., "Constraint equations: A concise compilable representation for quantified constraints in semantic networks," in *Proceedings of the National Conference on Artificial Intelligence*, AAAI, Washington, D. C., August 1984. Also available as USC/Information Sciences Institute, RS-84-137, June 1984.

12. MOSIS Project, "Chips and boards through MOSIS," in *Proceedings of COMPCON Spring '85*, pp. 184-186, IEEE, San Francisco, February 1985. Also available as USC/Information Sciences Institute, RS-85-149, February 1985.

13. Neches, R., W. Swartout, and J. Moore, "Enhanced maintenance and explanation of expert systems through explicit models of their development," in *Proceedings of the IEEE Workshop on Principles of Knowledge-Based Systems*, pp. 173-183, IEEE Computer Society, Denver, December 1984.

14. Neches, R., R. M. Balzer, N. Goldman, and D. Wile, "Transferring users' responsibilities to a system: The Information Management computing environment," in *INTERACT '84: First IFIP Conference on Human-Computer Interaction*, pp. 195-200, IFIP, London, September 1984.

15. Postel, J., "Internetwork applications using the DARPA protocol suite," in *INFOCOM '85, Computers and Communications Integration: The Confluence at Mid-Decade*, pp. 37-42, IEEE, Washington, D. C., March 1985. Also available as USC/Information Sciences Institute, RS-85-151, April 1985.

16. Sondheimer, N. K., R. M. Weischedel, and R. J. Bobrow, "Semantic interpretation using KL-ONE," in *Proceedings of Coling84: 10th International Conference on Computational Linguistics and 22nd Annual Meeting of the Association for Computational Linguistics*, pp. 101-107, ACL, Stanford, July 1984. Also available as USC/Information Sciences Institute, RR-85-154, April 1985.

17. Wilczynski, D., and N. Sondheimer, "Transportability in the Consul system: Model modularity and acquisition," in *Workshop on Transportable Natural Language Processing*, National Science Foundation, Durham, N.C., October 1984. Also available as USC/Information Sciences Institute, RR-85-154, April 1985.

Technical Reports

1. Brandan, D., J. Bentley, T. Gannon, M. Harrison, J. Rignati, F. Ris, and N. Sondheimer, *JTECH Panel Report on Computer Science in Japan*, Science Applications International Corporation, La Jolla, Calif., JTECH-TAR-8401, December 1984.
2. DeSchon, A., *MCI Mail/ARPA Mail Forwarding*, USC/Information Sciences Institute, RR-84-141, August 1984.
3. Kirton, P., *EGP Gateway Under Berkeley UNIX 4.2*, USC/Information Sciences Institute, RR-84-145, August 1984.
4. Mann, W. C., *Research on Knowledge Delivery*, USC/Information Sciences Institute, SR-84-148, December 1984.
5. Matthiessen, C. M. I. M., *Choosing Tense in English*, USC/Information Sciences Institute, RR-84-143, November 1984.

Informal Project Notes

1. Butler, M., J. Postel, D. Chase, J. Goldberger, and J. Reynolds, "Post Office Protocol - Version 2," USC/Information Sciences Institute, RFC 937, February 1985.
2. Finn, G., "Reliable Asynchronous Transfer Protocol (RATP)," USC/Information Sciences Institute, RFC 916, October 1984.
3. Postel, J., and J. Reynolds, "Domain Requirements," USC/Information Sciences Institute, RFC 920, October 1984.
4. Postel, J., "Domain Name System Implementation Schedule - Revised," USC/Information Sciences Institute, RFC 921, October 1984.
5. Postel, J., and J. Reynolds, "Assigned Numbers," USC/Information Sciences Institute, RFC 923, October 1984.
6. Postel, J., and J. Reynolds, "Official ARPA-Internet Protocols," USC/Information Sciences Institute, RFC 924, October 1984.
7. Postel, J., "Multi-LAN Address Resolution," USC/Information Sciences Institute, RFC 925, October 1984.
8. Postel, J., and J. Reynolds, "Internet Project Summary Report for October 1, 1983, through September 30, 1984," USC/Information Sciences Institute, November 1984.
9. Postel, J., "A DoD Statement on the NRC Report," USC/Information Sciences Institute, RFC 945, May 1985.
10. Reynolds, J., and J. Postel, "ARPA-Internet Protocol Policy," USC/Information Sciences Institute, RFC 902, July 1984.

11. Reynolds, J., "Post Office Protocol," USC/Information Sciences Institute, RFC 918, October 1984.
12. Reynolds, J., "ISI-Hosts," USC/Information Sciences Institute, October 1984.
13. Reynolds, J., and J. Postel, "Assigned Numbers," USC/Information Sciences Institute, RFC 943, April 1985.
14. Reynolds, J., and J. Postel, "Official ARPA-Internet Protocols," USC/Information Sciences Institute, RFC 944, April 1985.

RESEARCH AND ADMINISTRATIVE SUPPORT

Institute Administration:
Robert Blechen
- Judy Barboza
- Monica Boseman
- Barbara Davey
- Richard Dolen
- Diane Howell
- Gary Kennedy
- Alison La'Bat
- Gary Lum
- Gina Maschmeier
- Heidi Mucha
- Bob Pearson
- Annette Stoneking
- Calvin Sykes
- Dawn Thompson
- Karen Tonelli
- Lani Upton

Librarian:
Alicia Drake

Publications:
Victor Brown
Sheila Coyazo

Secretary to the Director:
Claire Jansa

Computing Facilities
Training and Information
Chloe Holg
Lisa Moses

Development Laboratory
Robert Parker
- Shorty Garza
- Bob Hines
- Jeff LaCoss
- Lee Magnone
- Rick Shiffman
- Jerry Wills